DETECCIÓN DE SOFT ERRORS EN BLOQUES MULTIPLICADORES DE FILTROS FIR

JUAN RABANILLO GARCÍA

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA, FACULTAD DE INFORMÁTICA,
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Ingeniería de Computadores

Directores:
Hortensia Mecha López
Pedro Reviriego Vasallo
Juan A. Maestro de la Cuerda

Julio 2013

Autorización de Difusión.

JUAN RABANILLO GARCÍA

Julio 2013

El abajo firmante, matriculado en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "**Detección de Soft Errors en Bloques Multiplicadores de Filtros FIR**", realizado durante el curso académico 2012-2013 bajo la dirección de Dª. Hortensia Mecha López y con la colaboración externa de dirección de D. Pedro Reviriego Vasallo y D. Juan Antonio Maestro de la Cuerda, en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano.

Este Trabajo Fin de Máster, que lleva por título: "Detección de Soft Errors en Bloques Multiplicadores de Filtros FIR" presenta un novedoso método para la detección de los fallos lógicos producidos en dichos circuitos electrónicos tras el impacto de partículas cósmicas, altamente energéticas.

El Método descrito en este trabajo de Máster en Investigación Informática, se basa en la inclusión de un nuevo componente en la estructura habitual de todo Bloque Multiplicador: el Registro Check, y en la utilización de Nodos sin compensación de errores.

El Registro Check aprovecha las precisas relaciones matemáticas que se establecen en todo Bloque Multiplicador de un Filtro FIR, de forma que a partir de los valores de entrada y salida de datos del Bloque, establece con seguridad plena si el Bloque Multiplicador ha operado correctamente o se ha producido algún Soft Error simple.

Para asegurar la ausencia de compensación de errores en los Bloques Multiplicadores se utiliza el Método de Replicación Parcial de nodos.

El Registro Check discrimina además si el Soft Error producido genera resultados inaceptables, que deben por tanto ser reprocesados y/o recalculados, o por el contrario solo ha alterado posiciones de los Nodos del Bloque Multiplicador, cuya influencia en el resultado final puede ser despreciada, al no afectar al nivel de precisión requerido en la fase de elección del Filtro FIR.

Los Bloques Multiplicadores obtenidos con este método presentan menor complejidad estructural que la asociada a la fabricación de circuitos ASIC con Redundancia Dual Modular.

El trabajo incluye el software específico para desarrollar el Método DetectError, implantándolo con facilidad en todo Bloque Multiplicador de un Filtro FIR.

Palabras clave.

Soft Error, Rayos Cósmicos, Filtros FIR Digitales, Bloque Multiplicador, Multiple Constant Multiplication (MCM), Grafos, Replicación de Nodos, Redundancia Dual Modular (DMR).

Resumen en ingles.

This thesis, entitled: "Detecting Soft Errors in FIR Filter Multiplier Blocks" presents a novel method for detecting logical faults in FIR filter electronic circuits caused by the impact of highly energetic cosmic particles.

The method described in this thesis for Master Degree in Computer Research, includes all the usual Multiplier Block structures, and a new register specially designed for this purpose, called Check Register. In addition requires compensating errors free Nodes.

This register uses the precise mathematical relationships established around each Multiplier Block for every FIR filter. So for a known input and output data in the filter, a simple Soft Error can be confirmed automatically without a doubt or on the contrary if Multiplier Block has operated correctly.

In order to assure no error compensation in Multiplier Blocks, Node Partial Replication Method will be used.

Check Register discriminates whether the produced Soft Error generates unacceptable results, which must therefore be reprocessed and / or recalculated. or conversely only modifies Multiplier Block node positions, whose influence on the final result, the output of the block, can be neglected as a result of not affecting the level of accuracy required in the FIR filter choice phase.

This new Multiplier Block requires less structural complexity than ASIC complexity associated to Dual Modular Redundancy (DMR) implementation.

It is included the specific software developed to provide FIR Filter Multiplier Blocks all the functionality described above.

Keywords.

Soft Errors, Cosmic Rays , Digital FIR Filter, Multiplier Block, Multiple Constant Multiplication (MCM), Graph theory, Node Replication, Dual Modular Redundancy (DMR).

Índice de contenidos.

Prefacio.

En este trabajo de Fin de Máster en Investigación Informática realizado en la Universidad Complutense de Madrid durante los cursos 2011-2012 y 2012-2013, que lleva por título: "Detección de Soft Errors en Bloques Multiplicadores de Filtros FIR", se presenta toda la secuencia de tareas realizadas y encaminadas al diseño de un nuevo tipo de Bloque Multiplicador.

A continuación se comentará brevemente cada una de ellas, remitiendo a los distintos capítulos que conforman esta Memoria para una descripción mucho más pormenorizada.

En el **capítulo 1** se detallan los antecedentes bibliográficos consultados para afianzar los conceptos, aplicaciones y técnicas utilizadas durante todo el trabajo, sin cuyo conocimiento habría sido imposible completarlo con éxito.

En el **capítulo 2**, tras analizar el funcionamiento de un Bloque Multiplicador estándar (BM estándar) y determinar sus propiedades más importantes, se propone como objetivo del presente trabajo, el diseño de un nuevo tipo de Bloque Multiplicador en el que no pueda producirse Compensación de Errores. Este nuevo tipo de Bloque se llamará BM DetectError.

Este objetivo lleva asociadas tres tareas adicionales.

La primera tarea consiste en el desarrollo del software específico, la aplicación DetectError, con la que sea posible diseñar Bloques Multiplicadores DetectError, utilizando como punto de partida los datos característicos de un Bloque Multiplicador estándar generado a través de la página Web de Spiral.

La segunda tarea del trabajo es presentar evidencias que confirmen la idoneidad del nuevo diseño de Bloque Multiplicador, estudiando para ello el comportamiento de Filtros FIR Paso bajo y Paso alto de orden 2 a 16, frecuencias normalizadas de corte 0'1 a 0'9 y precisión de cálculo desde 8 a 24 bit, de forma que sea posible comparar los resultados obtenidos con este nuevo diseño, con los proporcionados por otras opciones existentes: Filtro FIR con Bloque Multiplicador estándar y Filtro FIR con Redundancia Dual Modular provisto de BM estándar.

Por último, y como tercera tarea del trabajo, se confirmará mediante simulación digital con ModelSim que, en todas las situaciones de trabajo del Filtro FIR, el nuevo diseño de Bloque Multiplicador (BM DetectError) es capaz de detectar la aparición de cualquier Soft Error simple que pueda producirse.

En el **capítulo 3** se describe, en primer lugar, la estructura del nuevo tipo de Bloque Multiplicador, llamado **BM DetectError**, y se detalla las características diferenciales, que resuelven las graves deficiencias de los Bloques Multiplicadores estándar.

La primera de ellas es su capacidad para confirmar de forma inequívoca la aparición de cualquier Soft Error. El uso de BM DetectError obvia el empleo de Redundancia Dual Modular.

La segunda, y mucho más importante, es que la propia estructura del nuevo Bloque Multiplicador imposibilita la compensación interna de errores durante el cálculo. Por el contrario, cuando en un Bloque Multiplicador estándar existe compensación de errores, si se produce un Soft Error, es posible no detectarlo obteniendo la sensación engañosa de no haberse producido.

A continuación se presenta el software que, mediante técnicas de Replicación Parcial genera los nuevos Bloques Multiplicadores de Filtros FIR sin posibilidad de experimentar Soft Errors indetectables.

Se comenta cada una de las fases necesarias para implementar la funcionalidad deseada que, partiendo de los datos proporcionados por la Web Spiral, permite completar el diseño del nuevo Bloque Multiplicador DetectError y visualizar el grafo que describe su comportamiento.

En el **capítulo 4** se recogen todos los datos que evidencian la superioridad del Método DetectError frente a los otros métodos de detección de Soft Errors considerados, y con los que se confirma la consecución de la segunda tarea de este trabajo.

Teniendo en cuenta la relación directa existente entre el tamaño final de un circuito ASIC y su coste de fabricación, se determina el área necesaria del circuito lógico que implementa los distintos componentes: Bloque Multiplicador, Comparador y resto de Filtro FIR, para las tres opciones consideradas en el trabajo: Filtro FIR con Bloque estándar, Filtro FIR con Redundancia Dual Modular y Filtro FIR con Bloque Multiplicador DetectError.

Se confirma que la opción que incluye el Bloque Multiplicador DetectError ofrece mayores ahorros de superficie del circuito ASIC cuanto mayor es el requerimiento de precisión de cálculo requerida.

En el **capítulo 5**, se ratifica mediante simulación digital con la herramienta ModelSim que, en cualquier situación de trabajo del Filtro FIR, el nuevo diseño de Bloque Multiplicador DetectError, es capaz de detectar, sin lugar a dudas, la aparición de Soft Errors.

En el **capítulo 6**, se revisan los resultados y conclusiones finales del presente trabajo que permite anunciar la existencia del Método DetectError, método original para el diseño de nuevos Bloques Multiplicadores de Filtros FIR con capacidad para detectar la aparición de Soft Errors simples.

Este novedoso método asegura la detección de dichos errores de forma mucho más eficiente que con los métodos actualmente vigentes en el mercado.

Capítulo 1 - Trabajo bibliográfico previo.

Tengo seis honestos sirvientes
que me enseñaron todo lo que sé;
sus nombres son Qué y Por qué y Cuándo
y Cómo y Dónde y Quién.

"The elephant´s child" en Just So Stories for Little children
Rudyard Kipling, 1902

En este capítulo se recoge toda la información básica relacionada con el tema central de este trabajo de Fin de Máster, que he consultado personalmente antes de comenzar a planificar la línea de tareas necesarias para alcanzar con éxito el objetivo deseado.

Para su recopilación se ha aplicado la metodología de preguntas 5W+H.

Esta metodología se fundamenta en la aplicación de una batería de interrogaciones, a cada una de las Palabras Clave que acotan el tema de trabajo en cuestión:

¿Quién? (**W**ho),
¿Qué? (**W**hat),
¿Cómo? (**H**ow),
¿Cuándo? (**W**hen),
¿Dónde? (**W**here)
¿Por qué? (**W**hy),

La respuesta a cada una de dichas cuestiones nos indica claramente la información previa que debe ser recopilada y conocida para poder trabajar en ese tema.

En lo que se refiere a este proyecto toda esa información, se ha estructurado en los apartados que se describen en la figura 1.1.

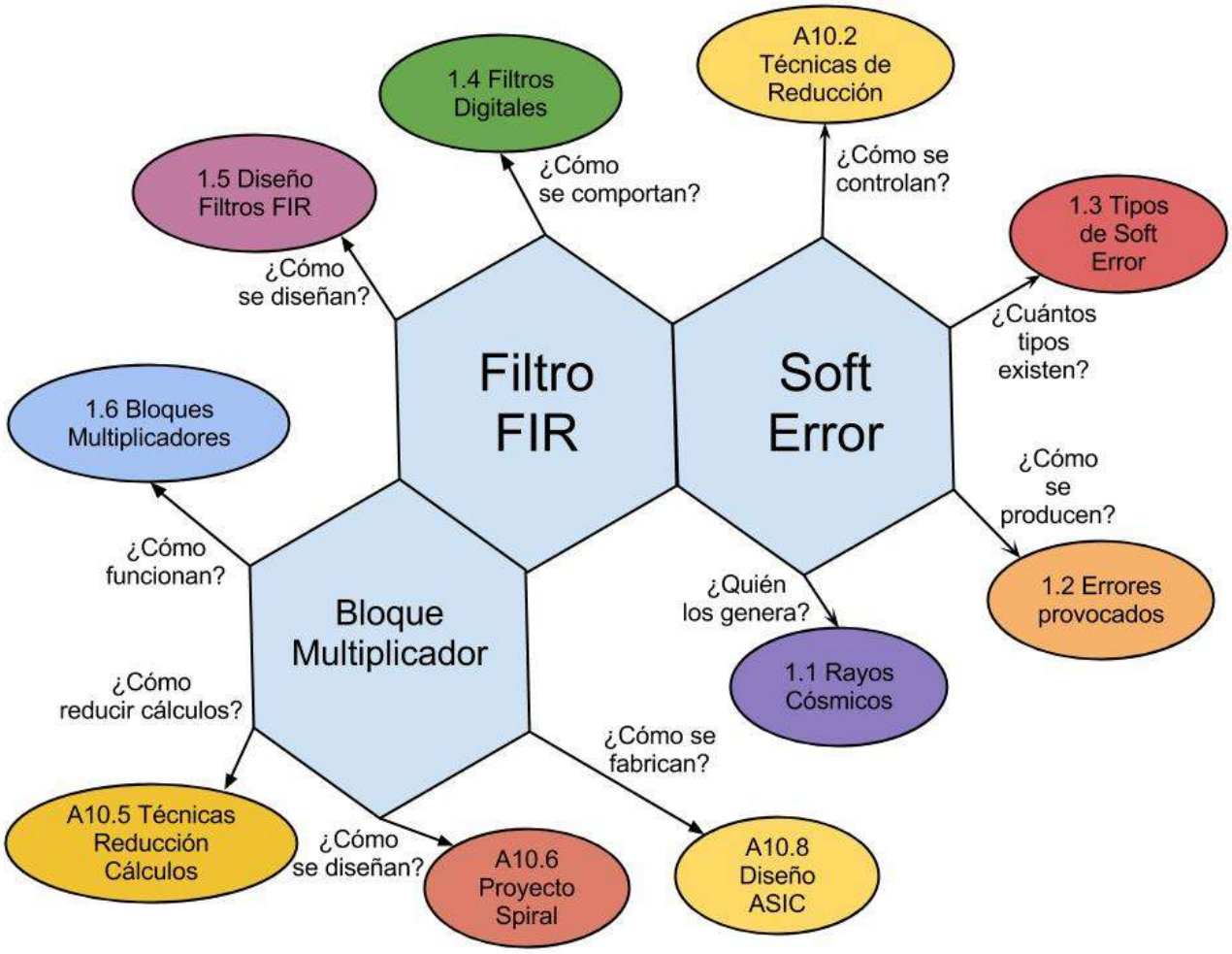


Figura 1.1 Diagrama 5W más H

1.1 Los Rayos Cósmicos.

En el año 1912 el físico estadounidense de origen austríaco Victor Franz Hess, [1], demostró que la ionización de la atmósfera terrestre aumentaba con la altitud, y concluyó por tanto que la radiación debía proceder del espacio exterior.

El hecho de que la intensidad de radiación dependiese de la altitud indicaba que las partículas que forman la radiación estaban eléctricamente cargadas y que eran desviadas por el campo magnético terrestre.

El físico R. A. Millikan acuñó para ello el término “Rayos Cósmicos”.

El origen de los Rayos Cósmicos no estuvo claro durante muchos años. Se sabía que el Sol emite Rayos Cósmicos de baja energía en los periodos en que se producen grandes erupciones solares, pero estos fenómenos estelares no son frecuentes; por lo tanto, no explicaban el origen de los rayos cósmicos, como tampoco lo explicaban las erupciones de otras estrellas semejantes al Sol.

En 2007, gracias al espectacular descubrimiento realizado por un grupo de científicos argentinos del Observatorio Pierre Auger [2], se inauguró una nueva rama de la Astronomía. Se encontraron evidencias de que la mayor parte de las partículas de Rayos Cósmicos proviene de la galaxia cercana, Centaurus-A. Esta galaxia, situada a unos 12 millones de años luz de la Tierra, también conocida como NGC 5128 [3], contiene un núcleo galáctico activo. Este núcleo activo se debe a la presencia de un agujero negro, probablemente supermasivo [4].

Al caer materia en la ergosfera del agujero negro con alta velocidad de rotación, parte de ella interacciona y escapa centrífugamente con gran energía, a enormes velocidades, en forma de protones y neutrones. Ver figura 1.2.

Los rayos cósmicos que alcanzan la atmósfera de la Tierra en su capa superior son principalmente (98%) protones y partículas alfa de alta energía. El resto son electrones y partículas pesadas ionizadas. A todas ellas se les llama partículas primarias.

Agujero negro con su disco de acreción

Esquema de un agujero negro

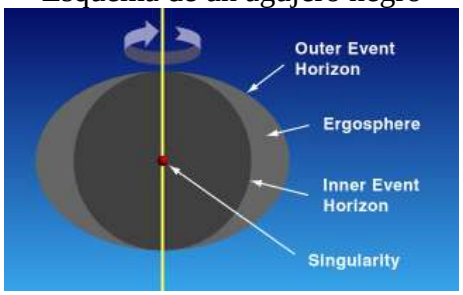


Figura1.2 Agujero negro: recreación y esquema

Cuando un rayo cósmico de alta energía llega a la atmósfera terrestre interactúa, tal y como se refleja en la figura 1.3, con los átomos que la forman, produciendo varios iones y partículas elementales entre los cuales se encuentran los piones, π^0, π^+ y π^-.

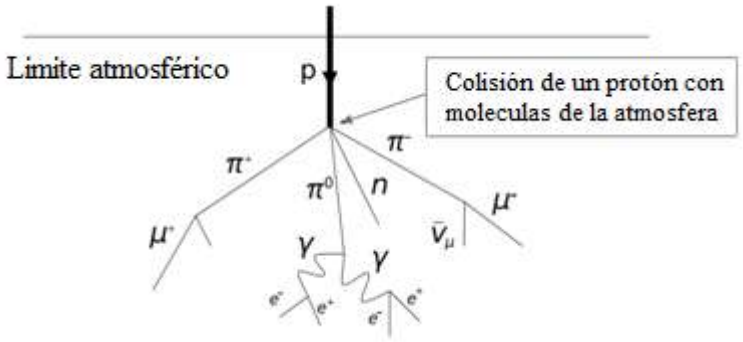


Figura 1.3 Cascada de Rayos Cósmicos

Cada pión π^0 decae rápidamente a dos rayos gammas que producirán nuevas cascadas electromagnéticas.

Cada uno de los rayos gamma generados produce un par electrón-positrón.

Estas partículas, por radiación de frenado (Bremsstrahlung) producen sucesivamente otros rayos gamma de energía más baja, que a su vez producen más pares en cadena hasta que la energía de las partícula es suficientemente baja (<80 MeV), momento en que aparecen otros procesos de disipación tales como el efecto Compton o la absorción fotoeléctrica. En general, los iones producidos tras el impacto inicial tienen todavía suficiente energía como para iniciar nuevas cascadas con otros átomos de la atmósfera creando nuevas partículas llamadas partículas secundarias.

Por su parte el pión π^- y el pión π^+ producen muones y antineutrinos, antimuones y neutrinos. Estas nuevas partículas producen otras partículas cuya carga eléctrica total es nula.

En definitiva, un solo rayo cósmico puede formar una cascada con más de 10^{11} partículas. En el punto 10.1 del Anexo A-10 se detalla el flujo de partículas según su contenido energético y datos relacionados.

1.2 Errores provocados por los Rayos Cósmicos.

El impacto sobre un circuito electrónico de cualquiera de las partículas altamente energéticas descritas en el punto anterior genera fallos lógicos en su funcionamiento.

Ello es debido a que la carga asociada a esas partículas, al atravesar los semiconductores de los circuitos electrónicos, invierte el estado de las puertas lógicas en que se encuentran implantados.

A continuación se describe con más precisión todo el proceso [5], que queda descrito gráficamente en las figuras 1.4 a 1.7.

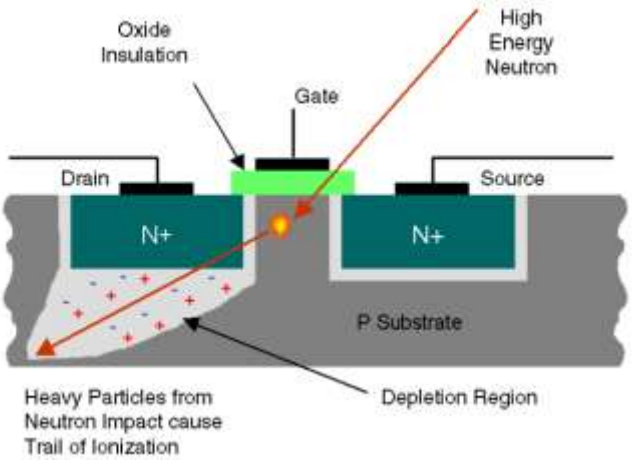


Figura 1.4 Colisión de una partícula cargada contra un circuito electrónico

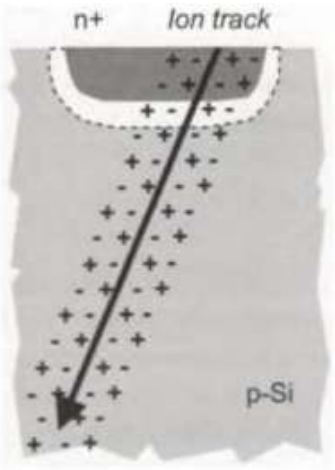


Figura 1.5 Entrada de partícula cargada

Entrada de la partícula cargada.

En el ejemplo estamos suponiendo que la entrada de la partícula se produce por la cara n+ y atraviesa la cara p de la unión.

A lo largo de la trayectoria se produce una densa distribución radial de pares de cargas positivas y negativas.

Si la pista de ionización resultante atraviesa la zona de deplexión, las cargas son rápidamente atrapadas por el campo eléctrico (en pocos picosegundos), neutralizando la carga existente en la unión, reduciendo permanentemente el voltaje si el nodo es flotante y temporalmente si es conducido.

En el exterior de la zona de deplexión, la distribución de carga que no está en equilibrio produce una distorsión en forma de túnel a lo largo de la trayectoria de la partícula, recolectando y redirigiendo la carga negativa por deriva.

Esta etapa dura entre 1 y 20 picosegundos.

Figura 1.6 Distribución de carga

Cuando el túnel de carga colapsa, tiene lugar una etapa de difusión de cargas hasta la unión, hasta que todas ellas son recogidas.

La duración de esta etapa supera generalmente los 20 picosegundos. Si la carga recogida por el nodo es suficiente, se supera el potencial de barrera y el estado del circuito cambia produciéndose un error.

Dado que esta situación no tiene lugar de forma continua, el error generado se denomina: SOFT ERROR o ERROR INTERMITENTE.

Figura 1.7 Difusión de carga

El SOFT ERROR fue observado por primera vez en chips de Intel en 1978 [6], quien comprobó la aparición de errores intermitentes en memorias DRAM (Dynamic Random Access Memory) y SRAM (Static Random Access Memory), que no tenían ninguna causa aparente.

La explicación del fenómeno fue descrita teóricamente por Ziegler y Lanford en 1979 [7].

Desde entonces, y como consecuencia de la reducción de tamaño de los componentes de los circuitos electrónicos utilizados, la frecuencia de error ha ido incrementándose de forma progresiva, haciendo necesario tomar medidas para atajar dicho problema [8].

1.3 Clases de Soft Error.

En la figura 1.8 se recogen todas las posibles formas en que puede concretarse un SOFT ERROR [9].

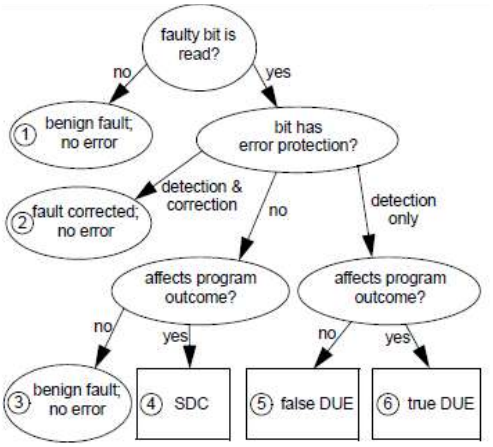


Figura 1.8 Clases de Soft Error

Fallo benigno (NO ERROR APARENTE).

Puede tener dos orígenes distintos, definidos en las etiquetas 1 y 3 de la figura 1.8.

En el primer caso, etiqueta 1, aunque se ha producido realmente un error no ha habido consecuencias por no haberse localizado ni utilizado dicha información.

En el segundo caso, etiqueta 3, el dato erróneo aun habiéndose localizado y procesado no ha afectado al programa y por tanto no ha generado ningún perjuicio.

Silent Data Corruption (SDC).

Indicado como etiqueta 4 en la figura 10.9, corresponde con un ERROR procesado que afecta negativamente al sistema.

Detected Unrecoverable Error o ERROR DUE.

Para evitar la aparición de Soft Errors, se ha utilizado una gran variedad de técnicas de protección de los circuitos electrónicos considerados críticos.

De entre todas ellas, la utilizada con más asiduidad es la denominada Redundancia Modular Múltiple. En dicha técnica, el circuito electrónico a proteger se replica, es decir, se repite varias veces.

Cuando la repetición es doble, REDUNDANCIA MODULAR DUAL (DMR), las salidas de los dos módulos idénticos se comparan y cuando hay diferencia se confirma la existencia de un

error, pero como no puede determinarse cuál de los dos módulos proporciona el valor verdadero, el error no es recuperable, Error DUE (Detected Unrecoverable Error).

Si dicho error afecta al resultado del programa estaríamos ante un verdadero error (TRUE DUE), etiqueta 6.

Cuando el error no afecta al resultado, etiqueta 5, se califica como FALSO ERROR (FALSE DUE).

Para confirmar la existencia de errores y poder corregirlos mediante el uso de técnicas de voto ponderado (etiqueta 2), es necesario efectuar, como mínimo, una réplica TRIPLE de los circuitos.

En ese caso estamos ante la llamada REDUNDANCIA TRIPLE MODULAR (TMR).

1.3.1 Cuantificación de la incidencia de Soft Errors.

Normalmente la industria indica los SOFT ERRORS de los circuitos comercializados con números SDC y DUE, cuya unidad de medida es el FIT (Failures in Time).

Un FIT representa la existencia de un error por cada mil millones de horas (10^9 horas) de trabajo.

La suma de números SDC y DUE se denomina SOFT ERROR RATE (SER) de un chip.

En muchas ocasiones se utiliza una unidad distinta al FIT, la llamada MTTF (Mean Time To Failure), inversa del FIT, y mucho más intuitiva.

La equivalencia entre ambas es inmediata. Una tasa de error de 1000 FIT es equivalente a una MTTF de 114 años $\left(\frac{10^9 h}{1000 \text{ errores } * 24\frac{\text{h}}{\text{dia}} * 365\frac{\text{dia}}{\text{año}}}\right)$.

En el punto 10.2 del Anexo A-10 de describen las técnicas existentes para reducir la magnitud de Soft Errors.

1.4 Filtros Digitales.

Este tipo de filtros tiene como entrada una secuencia discreta de datos y como salida otra secuencia discreta, que habrá experimentado ciertas variaciones en amplitud y/o fase dependiendo de la acción del filtro empleado. Ver figura 1.9.

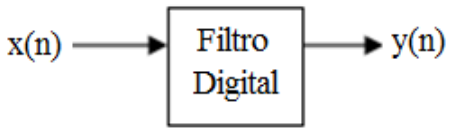


Figura 1.9 Esquema de un filtro digital.

1.4.1 Tipos de Filtros Digitales:

Filtros IIR, Infinite Impulse Response o Respuesta infinita al impulso.

Se trata de un tipo de Filtros Digitales en el que, como su nombre indica, si la entrada es una señal impulso, la salida tendrá un número infinito de términos no nulos, es decir, nunca vuelve al reposo.

La salida de los Filtros IIR depende de las entradas actuales y pasadas, y además de las salidas en instantes anteriores. Esto se consigue mediante el uso de realimentación de la salida.

Estructura de un Filtro IIR de forma directa Tipo I.

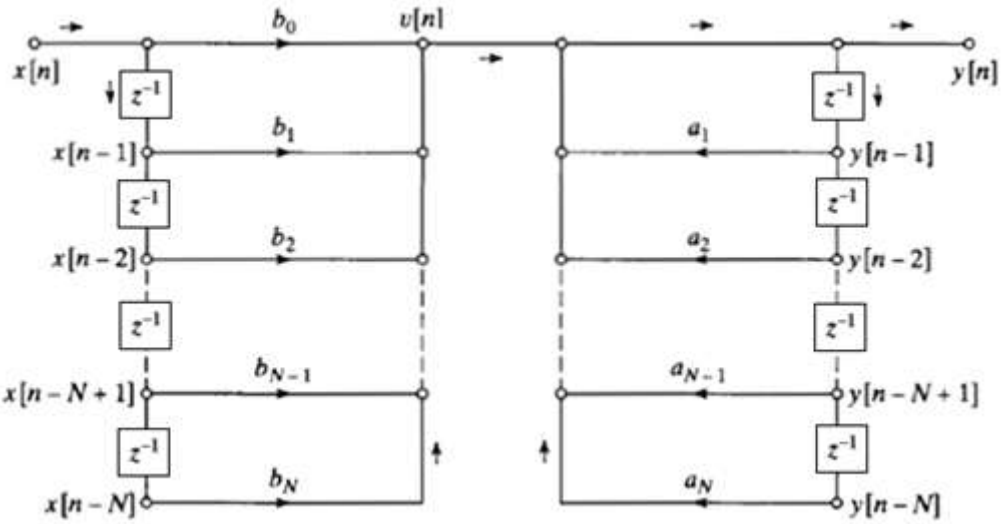


Figura 1.10 Filtro IIR forma directa Tipo I

En la estructura anterior se ha utilizado la simbología indicada en la figura 1.11.

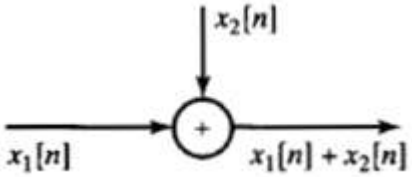
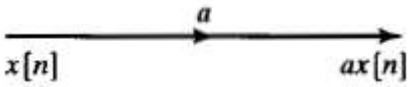
	**Suma de dos secuencias.**
	Dirección de flujo de señal
	Multiplicación de una secuencia por una constante.
	Inducción de un Retraso. Ver NOTA.

Figura 1.11 Simbología representación filtros

NOTA: En implementaciones digitales, la inducción de un retraso se lleva a cabo con la utilización de un registro de almacenamiento para cada línea del filtro.

Filtros FIR, Filtros de Respuesta Finita al Impulso o Finite Impulse Response.

Los Filtros Digitales de Respuesta Impulsional Finita (Finite Impulse Response) se basan en obtener la salida a partir, exclusivamente, de las entradas actuales y anteriores.

Dependiendo de la estructura de construcción, se conocen los dos tipos fundamentales de Filtros FIR indicados, respectivamente, en las figuras 1.12 y 1.13: Forma directa y Forma traspuesta:

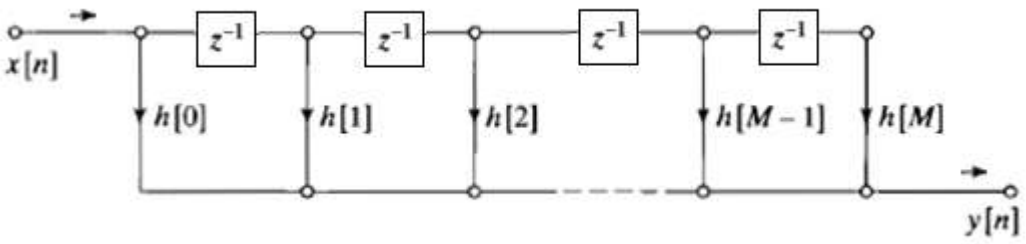


Figura 1.12 FIR Forma directa

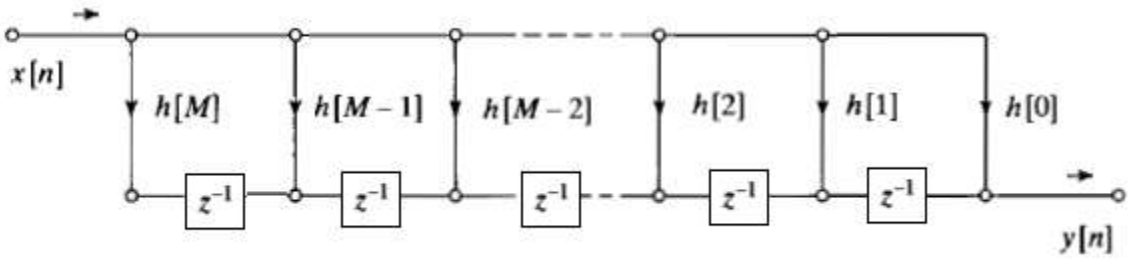


Figura 1.13 FIR Forma traspuesta

1.4.2 Respuestas de frecuencia de Filtros FIR Ideales.

Dependiendo del número de coeficientes y del tipo de simetría de un Filtro FIR existen las posibilidades recogidas en la Tabla 1.1.

Tabla 1.1 Tipos de Filtros FIR

Tipo:	Número de términos	Simetría
I	Impar	Simétrico h(k) = h(M-k)
II	Par	Simétrico h(k) = h(M-k)
III	Impar	Antisimétrico h(k) = - h(M-k)
IV	Par	Antisimétrico h(k) = - h(M-k)

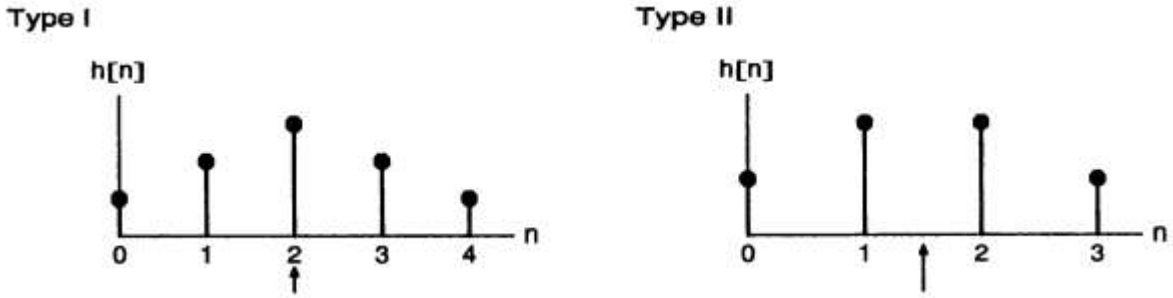


Figura 1.14 Filtros FIR tipos I y II

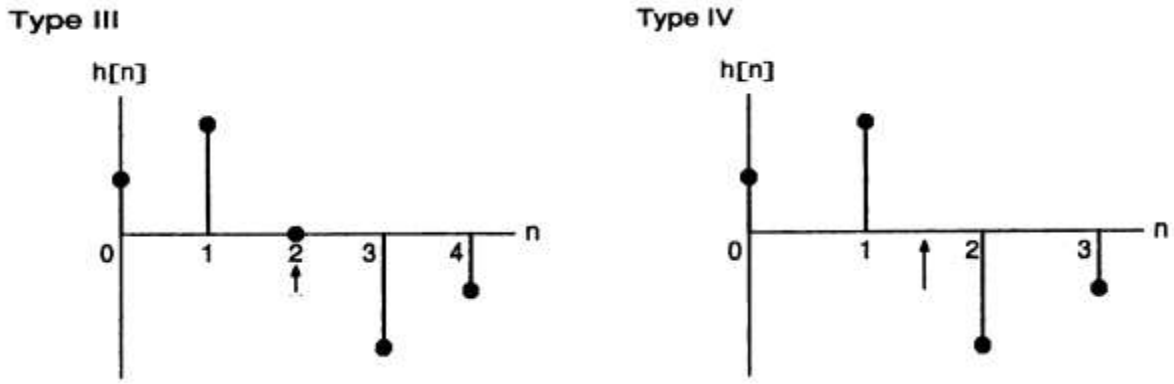


Figura 1.15 Filtros FIR tipos III y IV

En el punto 10.3 del Anexo A-10 se desarrollan las ecuaciones que describen la respuesta en frecuencia para cada uno de estos tipos de filtro.

1.4.3 Tipos de filtrado ideal según respuesta deseada al impulso.

Se aprovecha el hecho de que la respuesta en frecuencia de un filtro $H_D(\omega)$ y la correspondiente respuesta al impulso, $h_D(n)$ se relacionan por la transformada inversa de Fourier.

$$\boldsymbol{h_D(n) = \left(\frac{1}{2\pi}\right) \int_{-\pi}^{\pi} H_D(\omega) e^{j\omega n}\ d\omega} \quad (\mathbf{1})$$

El subíndice D se usa para indicar que se trata de la respuesta al impulso de un filtro ideal.

Existen 4 posibles respuestas al impulso de un filtro ideal, según la opción de funcionamiento del filtro:

- Filtrado Paso bajo.
- Filtrado Paso alto.
- Filtrado Paso banda.
- Filtrado Para banda.

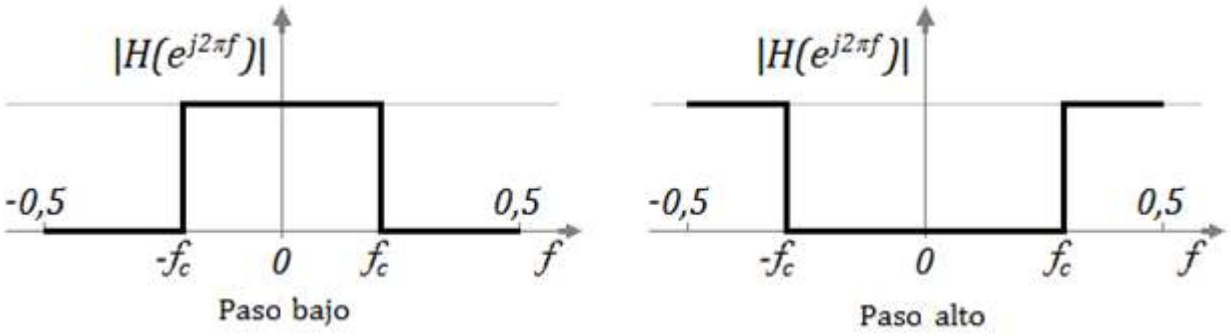


Figura 1.16 Respuesta Filtros Paso bajo y Paso alto

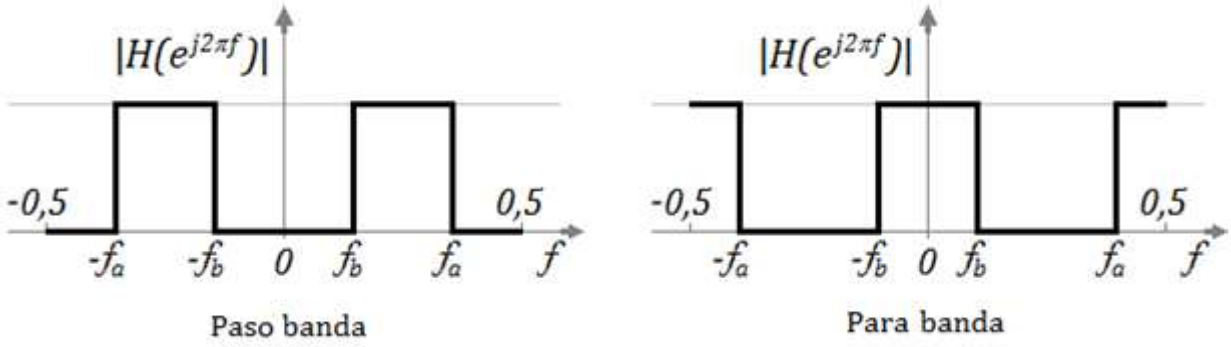


Figura 1.17 Respuesta Filtros Paso banda y Para banda

Siendo:

f_c: Frecuencia de corte normalizada

f_b: Frecuencia de corte baja normalizada

f_a: Frecuencia de corte alta normalizada

1.4.4 Diseño de Filtros FIR Reales.

Existen tres grandes bloques de métodos de diseño de Filtros FIR con fase lineal: El método de la Ventana, el muestreo en Frecuencia y el método del Rizado constante (equiripple).

El método de la Ventana se basa en acotar la respuesta impulsional infinita de un filtro ideal, el método del muestreo en Frecuencia propone que se fijen una serie de puntos de la respuesta en frecuencia del sistema y, a partir de la DFT inversa, obtener los coeficientes del filtro. Por último existe una familia de métodos que se basan en definir la respuesta en frecuencia ideal del filtro y, fijado un orden, obtener los coeficientes que generen la respuesta más aproximada, en particular, los más comunes se basan en la aproximación de Tchebyshev

Dado el alcance del presente trabajo, solo describiremos someramente el Método de la Ventana, habida cuenta de que será el empleado para generar los Filtros FIR utilizados en la parte experimental de este trabajo

El método se basa en que cuando queremos implementar, por ejemplo, un filtro Paso bajo con una respuesta ideal, que implica una transición muy abrupta de la banda pasante a la atenuada, como la respuesta impulsional es infinita y no causal, para obtener un filtro FIR realizable se debe truncar la respuesta h(n) y retardarla hasta convertirla en causal.

A grandes rasgos, el procedimiento consta de las siguientes etapas:

a) Obtener la respuesta impulsional del tipo de filtrado ideal que deseamos diseñar.

Esta etapa se ha descrito en el punto anterior.

b) Truncar (***enventanar***) la respuesta impulsional del filtro ideal mediante el uso de una Ventana.

Esta operación consiste en multiplicar la respuesta impulsional del filtro por la respuesta impulsional de una ventana:

$$\boxed{\boldsymbol{h(n) = h_i(n) * w(n)}}$$

Y posteriormente realizar la convolución del producto.

$$\boldsymbol{H(e^{j\omega}) = \left(\frac{1}{2\pi}\right) \int_{-\pi}^{\pi} H_D(e^{j\theta}) W e^{j(\omega-\theta)}\, d\theta}$$

La respuesta impulsional de la ventana, w(n), es de la siguiente forma:

$$w(n) = \begin{matrix} funcion\ simétrica & para\ 0 \le n \le M \\ 0 & en\ el\ resto\ situaciones \end{matrix}$$

En el Anexo punto 10.4 del Anexo A-10 se detallan las características de las ventanas más utilizadas.

c) Desplazar la respuesta impulsional enventanada (retrasar) un número adecuado de muestras para hacerla causal.

También se puede desplazar la respuesta impulsional del filtro ideal previamente, para que la secuencia enventanada sea causal.

En la figura 1.18 se recoge cualitativamente el resultado de aplicar una ventana a un filtro Paso bajo ideal.

Como puede comprobarse el efecto suaviza la pendiente de la transición de banda.

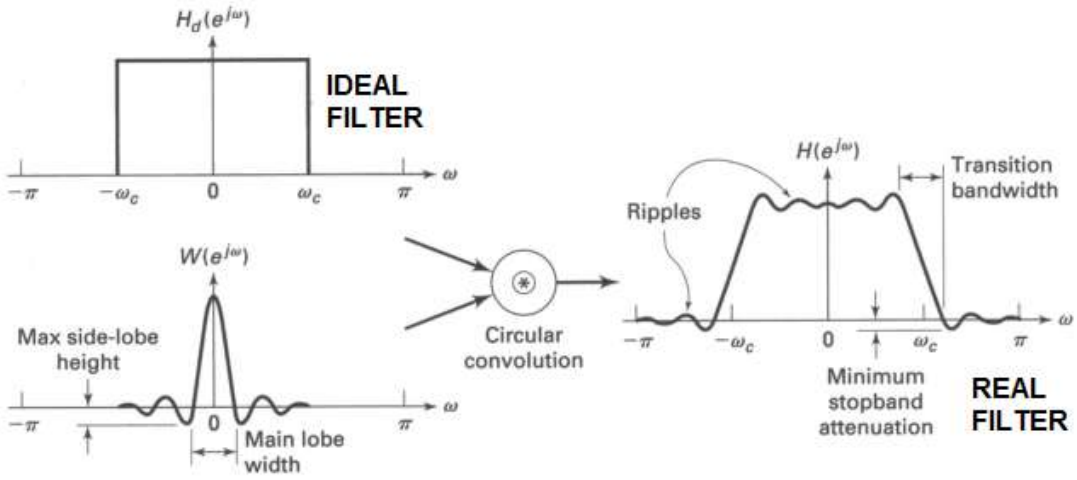


Figura 1.18 Respuesta de un filtro real

En las figuras 1.19 a 1.22 se representa, para los diferentes tipos de filtrado reales posibles, la Magnitud de la respuesta frente a la frecuencia.

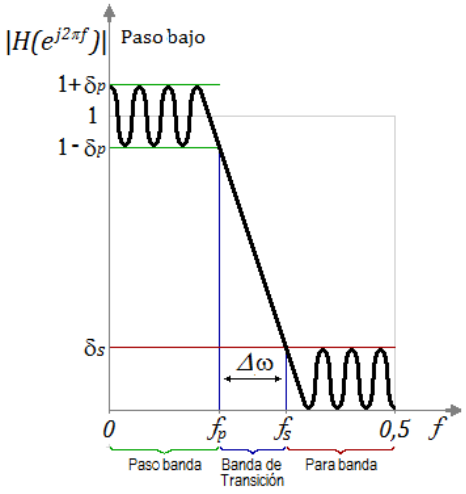


Figura 1.19 Respuesta filtro Paso bajo

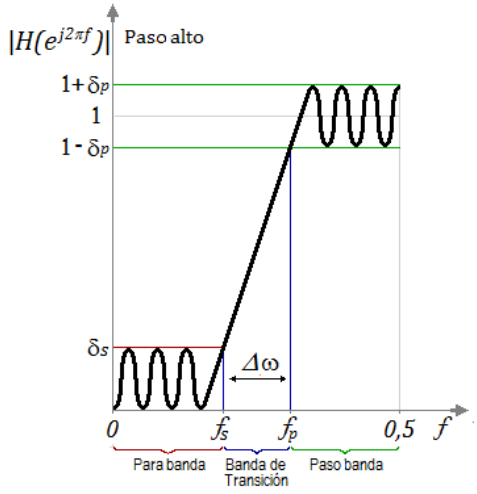


Figura 1.20 Respuesta filtro Paso alto

$$f_c = f_p + \Delta\omega/2 = \\ = f_p + (f_s - f_p)/2 = (f_s + f_p)/2$$

$$f_c = f_p - \Delta\omega/2 = \\ = f_p - (f_p - f_s)/2 = (f_p + f_s)/2$$

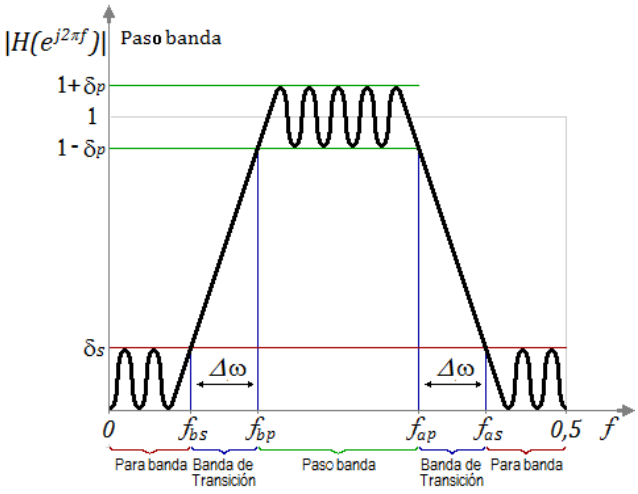


Figura 1.21 Respuesta filtro Paso banda

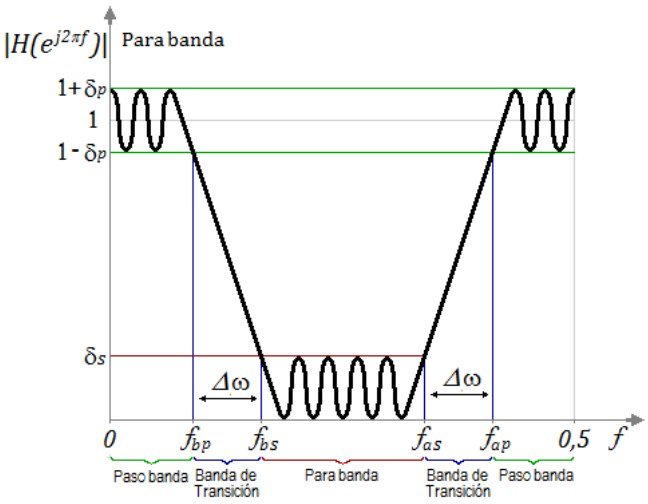


Figura 1.22 Respuesta filtro Para banda

$$f_b = f_{bp} - \Delta\omega/2 =$$
$$= f_{bp} - (f_{bp} - f_{bs})/2 =$$
$$= (f_{bp} + f_{bs})/2$$

$$f_a = f_{ap} + \Delta\omega/2 =$$
$$= f_{ap} + (f_{as} - f_{ap})/2 =$$
$$= (f_{ap} + f_{as})/2$$

$$f_a = f_{ap} - \Delta\omega/2 =$$
$$= f_{ap} - (f_{ap} - f_{as})/2 =$$
$$= (f_{ap} + f_{as})/2$$

$$f_b = f_{bp} + \Delta\omega/2 =$$
$$= f_{bp} + (f_{bs} - f_{bp})/2$$
$$= (f_{bp} + f_{bs})/2$$

NOTA: f_c es la frecuencia de corte del Filtro, f_p es la frecuencia paso banda y f_s es la frecuencia para banda.

1.5 Diseño de Filtros FIR con MATLAB.

MATLAB (abreviatura de MATrix LABoratory) es el software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

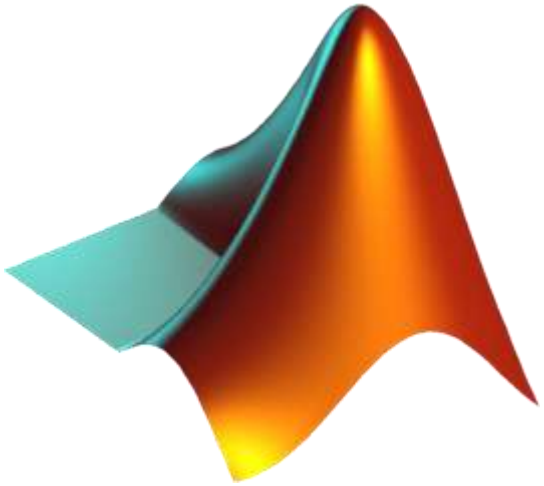


Figura 1.23 Logo de MATLAB

Este software se encuentra disponible para las plataformas Unix, Windows y Mac OS X.

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware.

El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber:

Simulink (plataforma de Simulación Multidominio)

GUIDE (editor de interfaces de usuario - GUI).

Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Las funcionalidades de MATLAB se agrupan en más de 35 cajas de herramientas y paquetes de bloques (para Simulink), clasificadas en las categorías indicadas en la tabla 1.2.

Tabla 1.2 Herramientas y paquetes de bloques

MATLAB (Cajas de herramientas)	Simulink
Matemáticas y Optimización	Modelado de punto fijo
Estadística y Análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de Imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL
Desarrollo de aplicaciones	Tarjetas integradas
Informes y conexión a bases de datos	Verificación, validación y comprobación

Dentro del Toolbox para procesado de Señales y comunicaciones para DSP, se encuentra el módulo FILTER DESIGN TOOLBOX, con el que puede, de forma muy rápida e intuitiva, diseñar cualquier tipo de Filtro digital.

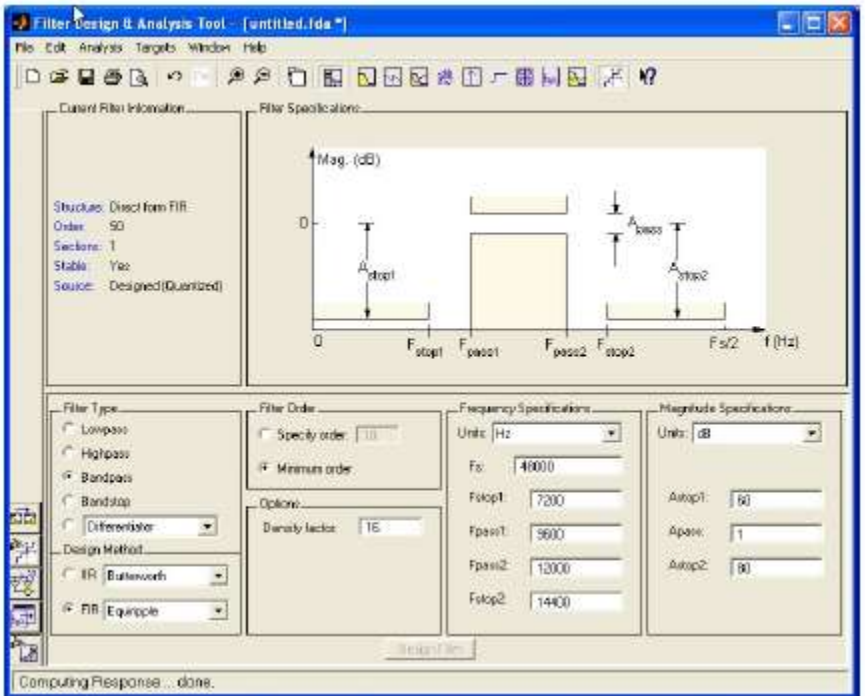


Figura 1.24 Pantalla de acceso al módulo Fdatool (Análisis y diseño de filtros)

1.5.1 Función MATLAB para cálculo de coeficientes de Filtros FIR Reales.

La función **fir1(N, Wn, TipoFiltrado, Ventana),** proporciona los coeficientes de los Filtros FIR Reales, que usaremos en este trabajo.

N Número de Orden del Filtro.

Cuando el tipo de filtrado es Paso alto o Para banda y el Orden del filtro es impar, la función calcula el Filtro de Orden N+1. Por ello en este trabajo solo se calcularán los Filtros Paso alto de orden par.

Wn Frecuencia normalizada de corte del filtro. ($0 \leq Wn \leq 1$).

$Wn = \frac{2 * \text{Frecuencia de corte}}{\text{Frecuencia de muestreo}}$.

En caso de diseñar Filtrado Para banda o Paso banda se debe indicar las frecuencias normalizadas borde inferior y superior Para banda y Paso banda con el formato:
[W1 W2].

Ventana La Ventana por defecto es del tipo Hamming, pero se puede indicar cualquiera de las otras descritas en el punto 10.4 del Anexo A-10: Rectangular, Barlett o Hanning.

Tipo de Filtrado El filtrado Paso bajo es el tipo de filtrado por defecto, pero se puede indicar cualquiera de los existentes:

Paso alto: 'high'
Paso banda: 'bandpass'
Para banda: 'stop'

Sintaxis: B = **fir1(N, Wn, Ventana, TipoFiltrado).**

El sistema proporciona la matriz B que contiene N+1 coeficientes.

En la tabla 1.3 se presenta, a modo de ejemplo, los coeficientes de un Filtro FIR Paso banda de Orden 64, para filtrado en el intervalo de frecuencias normalizadas comprendido entre 0'4 y 0'6

Tabla 1.3 Coeficientes Filtro FIR Paso banda Orden 64

Coeficientes B(ij) de un Filtro FIR Paso banda de Orden 64 Frecuencias normalizadas de corte [0'4 0'6]							
j \ i	0	1	2	3	4	5	6
0	0.0005	-0.0024	0.0112	-0.0927	-0.0202	0.0047	-0.0008
1	-0.0003	0.0015	-0.0068	0.0981	0.0091	-0.0021	0.0003
2	0.0000	-0.0000	0.0000	0.8995	0.0000	-0.0000	0.0000
3	0.0003	-0.0021	0.0091	0.0981	-0.0068	0.0015	-0.0003
4	-0.0008	0.0047	-0.0202	-0.0927	0.0112	-0.0024	0.0005
5	0.0013	-0.0075	0.0329	0.0841	-0.0133	0.0028	
6	-0.0018	0.0102	-0.0465	-0.0730	0.0136	-0.0027	
7	0.0023	-0.0124	0.0602	0.0602	-0.0124	0.0023	
8	-0.0027	0.0136	-0.0730	-0.0465	0.0102	-0.0018	
9	0.0028	-0.0133	0.0841	0.0329	-0.0075	0.0013	

Con estos coeficientes, MATLAB nos permite representar fácilmente la respuesta del filtro considerado a lo largo de todo el intervalo de frecuencias normalizada, comprobándose, que en efecto, la respuesta reflejada en la figura 1.25 corresponde al Filtro FIR Paso banda que deseábamos diseñar.

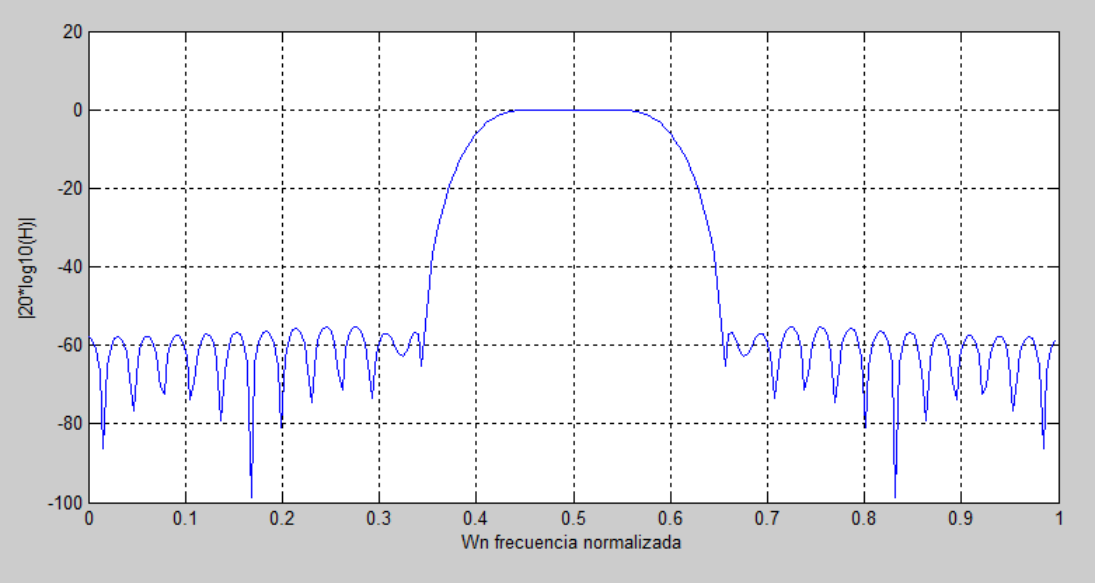


Figura 1.25 Respuesta del Filtro FIR Paso banda Orden 64

En el Anexo A-1.1, se encuentran recogidos los coeficientes de Filtros FIR Paso bajo calculados con MATLAB para Filtros de Orden 2 a 16 y diseñados para trabajar con frecuencias normalizadas de corte comprendidas entre 0’1 y 0’9.

1.6 Bloques Multiplicadores de Filtros Digitales.

En todos los Filtros Digitales se lleva a cabo una serie de operaciones matemáticas, sumas y multiplicaciones en las que están implicados la señal de entrada y los coeficientes que caracterizan el Filtro.

Así, para un Filtro FIR en su forma traspuesta, las operaciones matemáticas tienen lugar en cada uno de los Bloques Aritméticos del Filtro, cuya estructura se representa en la figura 1.26, uno para cada uno de los distintos elementos que caracteriza el modo de acción de dicho Filtro.

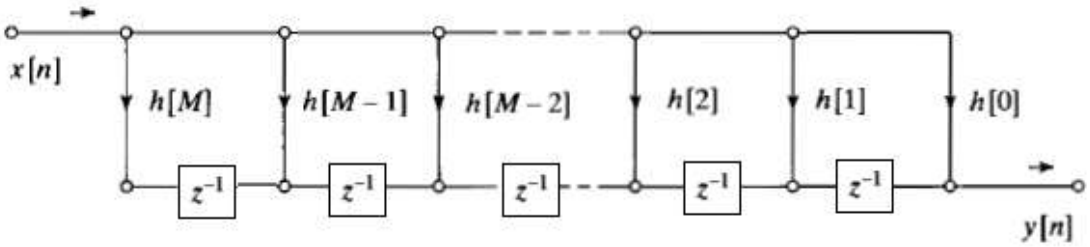


Figura 1.26 Estructura filtro FIR forma traspuesta

Ello da lugar a la realización de M multiplicaciones por cada entrada digital al filtro, siendo M = Orden del filtro + 1, y requiriendo por tanto un elevado coste computacional y una redundancia de orden M de los componentes implicados en dichas operaciones.

Para minimizar este coste computacional se utiliza la estrategia Multiple Constant Multiplication (MCM). En el punto 10.5 del Anexo A-10 se procede a detallar su funcionamiento y se describe su origen.

Generadores Spiral on-line para Multiple Constant Multiplication.

El Proyecto Spiral, detallado en el punto 10.6 del Anexo A-10, permite generar on-line Bloques Multiplicadores, para llevar a cabo de forma eficiente las operaciones de Multiple Constant Multiplication.

Los dos generadores disponibles se indican en la figura 1.27:

- Parallel Multiple Constant Multiplication (MCM)
- Multiplexed Multiple Constant Multiplication (MUX-MCM)

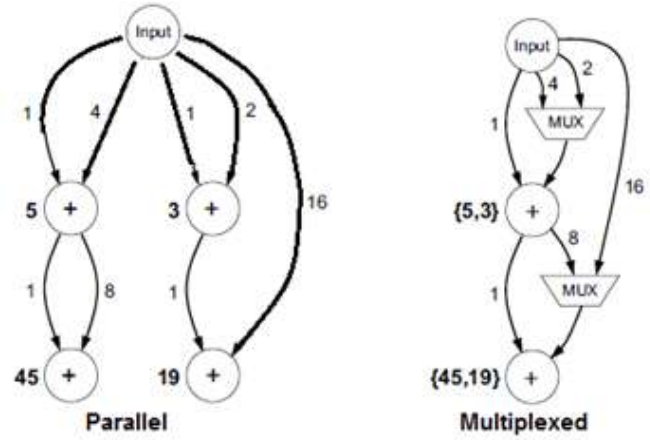


Figura 1.27 Opciones para realizar MCM

En este proyecto utilizaremos el Bloque Multiplicador MCM Parallel.

1.6.1 Operativa para el cálculo de un Bloque Multiplicador MCM Parallel.

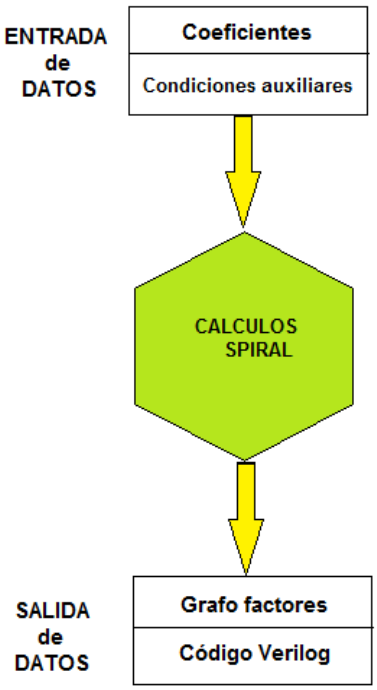


Figura 1.28 Diagrama de flujo operativo MCM

El generador on-line proporcionado por Spiral responde al esquema operativo de la figura 1.28:

a) Elección del orden filtro FIR a estudiar e introducción de parámetros descriptivos.

b) Cálculos internos realizados con el algoritmo Hcub implementado en Spiral.

c) Salida de datos:

Grafo que contiene los factores del Bloque Multiplicador y las conexiones implicadas entre nodos del MCM.

Código Verilog, que contiene el diseño del Módulo Multiplicador.

Elección del Filtro FIR deseado e introducción de los parámetros de diseño.

Aunque el Menú principal presenta la expresión general de un filtro IIR:

$$a_0 y[n] = \sum_{k=0}^{N-1} b_k x[n-k] - \sum_{k=1}^{N-1} a_k y[n-k]$$

Standard Difference Equation

El generador on-line incorpora el criterio interno de que cuando el coeficiente $a_0 = 1$ y el resto de los coeficientes a_k son todos ceros, la expresión general a tratar es:

$$y[n] = \sum_{k=0}^{N-1} b_k x[n-k]$$

Standard Difference Equation

Que representa el comportamiento de un filtro FIR.

En la figura 1.29 se recoge la estructura del Menú Web para introducir los distintos parámetros de diseño:

Example filter:

Se encuentran implementadas tres opciones:

Filtro diseñado por el usuario (Custom filter).

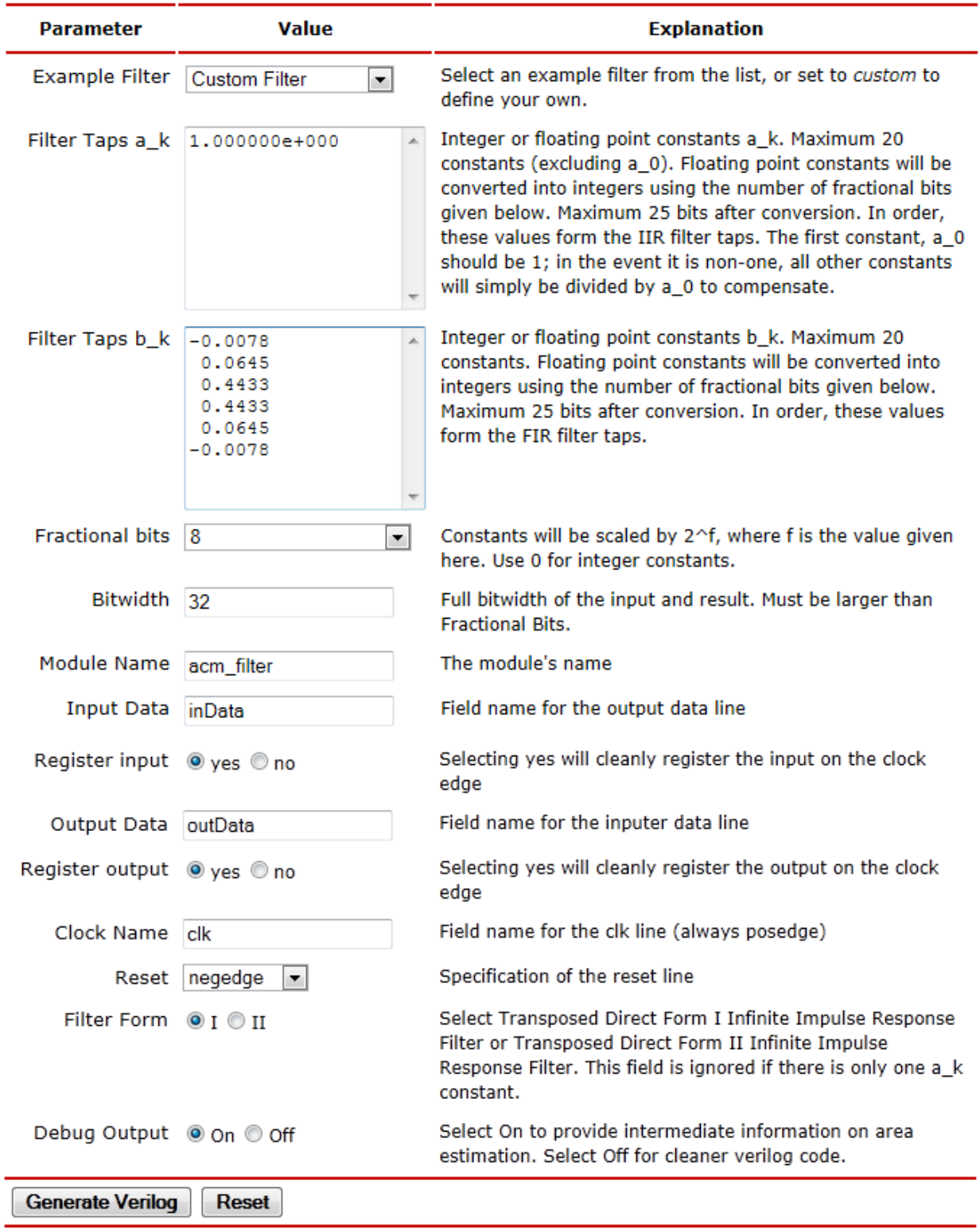
Filtro Paso bajo (con 10 opciones preestablecidas).

Filtro Paso alto (con 10 opciones preestablecidas).

En este trabajo se ha utilizado Filtros FIR Paso bajo y Paso alto diseñados con el concurso de MATLAB.

Filter taps a_k

Solo debe figurar el valor 1, para que, tal y como se indicó anteriormente, los cálculos representen a un filtro FIR.



Parameter	Value	Explanation
Example Filter	Custom Filter	Select an example filter from the list, or set to *custom* to define your own.
Filter Taps a_k	1.000000e+000	Integer or floating point constants a_k. Maximum 20 constants (excluding a_0). Floating point constants will be converted into integers using the number of fractional bits given below. Maximum 25 bits after conversion. In order, these values form the IIR filter taps. The first constant, a_0 should be 1; in the event it is non-one, all other constants will simply be divided by a_0 to compensate.
Filter Taps b_k	-0.0078 0.0645 0.4433 0.4433 0.0645 -0.0078	Integer or floating point constants b_k. Maximum 20 constants. Floating point constants will be converted into integers using the number of fractional bits given below. Maximum 25 bits after conversion. In order, these values form the FIR filter taps.
Fractional bits	8	Constants will be scaled by 2^f, where f is the value given here. Use 0 for integer constants.
Bitwidth	32	Full bitwidth of the input and result. Must be larger than Fractional Bits.
Module Name	acm_filter	The module's name
Input Data	inData	Field name for the output data line
Register input	yes no	Selecting yes will cleanly register the input on the clock edge
Output Data	outData	Field name for the inputer data line
Register output	yes no	Selecting yes will cleanly register the output on the clock edge
Clock Name	clk	Field name for the clk line (always posedge)
Reset	negedge	Specification of the reset line
Filter Form	I II	Select Transposed Direct Form I Infinite Impulse Response Filter or Transposed Direct Form II Infinite Impulse Response Filter. This field is ignored if there is only one a_k constant.
Debug Output	On Off	Select On to provide intermediate information on area estimation. Select Off for cleaner verilog code.

Generate Verilog Reset

Figura 1.29 Captura Menú Web Spiral

Filter taps b_k

Los coeficientes del filtro Paso bajo de N elementos, se calculan con el concurso de MATLAB. El número máximo de coeficientes queda limitado a 20.

Cuanto mayor es el número de coeficientes, más complejo es el grafo del Bloque Multiplicador. El grafo se dilata en sentido horizontal, En efecto es necesario generar más salidas.

En la figura 1.30 se representan los grafos del Bloque Multiplicador de un Filtro FIR Paso bajo de orden 3 y orden 8 con frecuencia de corte normalizada de 0'6, y precisión de 8 bits. Como puede comprobarse, el grafo de orden 3 tiene 4 salidas y 3 nodos mientras que el grafo de orden 8 requiere 9 salidas y 9 nodos.

Fraccional bits:

Se han utilizado cinco supuestos: 8, 12, 16, 20 y 24 bits de precisión.

Cuanto mayor es el número de bits de precisión exigido, más complejo es el grafo del Bloque Multiplicador. En efecto es necesario utilizar nodos con números enteros que representan un factor de multiplicación mucho mayor.

El grafo se dilata en sentido vertical, es decir, cada entrada debe atravesar mayor número de nodos para generar la correspondiente salida.

En la figura 1.31 se representan los grafos del Bloque Multiplicador de un Filtro FIR Paso bajo de orden 3 y frecuencia de corte normalizada de 0'5, con precisión de 8 y 16 bit. Como puede comprobarse, el grafo con precisión 16 bit tiene 18 nodos mientras que solo son necesarios 5 para conseguir precisión 8 bit.

Bitwidth:

Se ha utilizado en todo momento el valor preestablecido por Spiral de 32 bits, con lo que los nodos tendrán por defecto un ancho total de treinta y dos bit por encima de la precisión deseada.

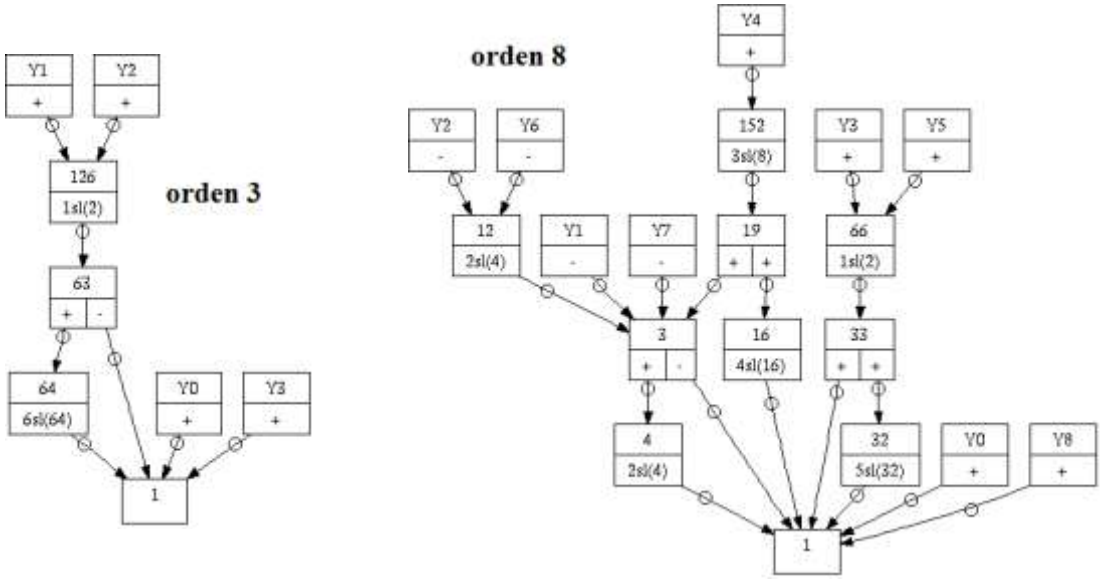


Figura 1.30 Grafos Filtro FIR Paso bajo precisión 8 bit, con orden 3 y 8 Frecuencia 0'6

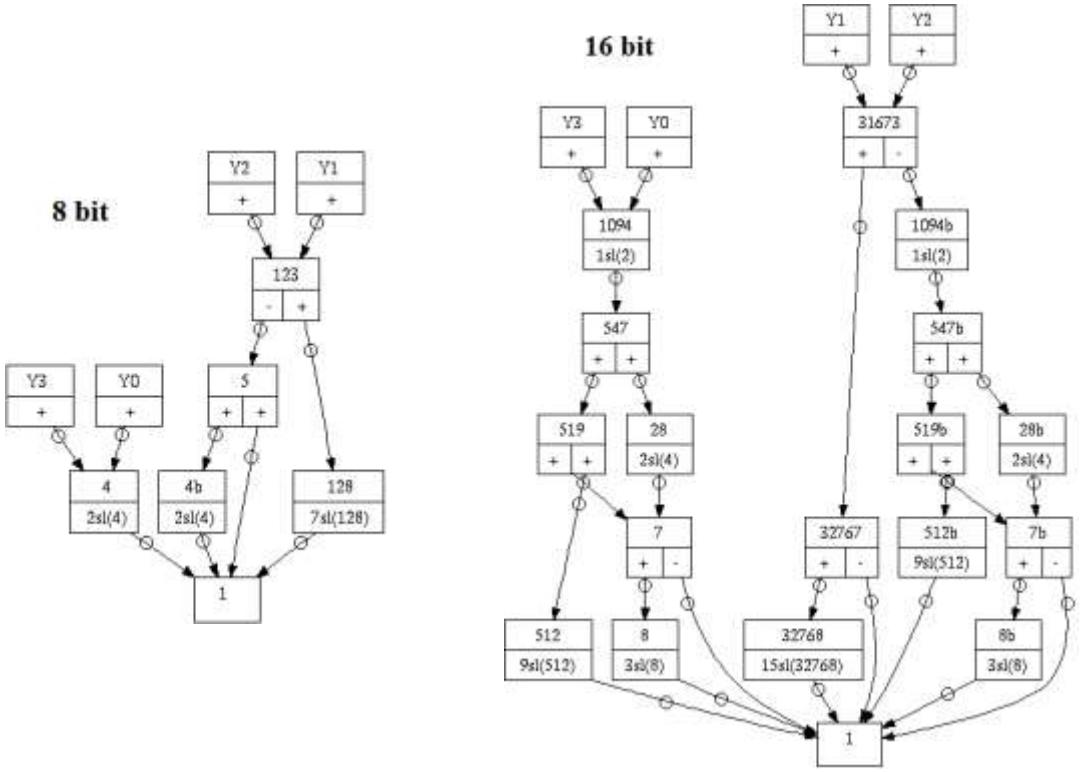


Figura 1.31 Grafos Filtro FIR Paso bajo orden 3, con precisión 8 y 16 bit Frecuencia 0'5

Etapa de Cálculo.

Es totalmente invisible al usuario. El generador utiliza el algoritmo Hcub, detallado en el punto 10.5.1 del Anexo A-10, para determinar el grafo del Bloque Multiplicador.

El primer paso consiste en transformar los coeficientes del filtro introducidos en el menú online, con valores decimales entre 0 y 1, a valores enteros redondeados comprendidos en el rango 0 a $2^{\text{fractional-bit}}$.

Así para el Filtro FIR Paso bajo de orden 5, Frecuencia normalizada 0'5 y precisión de cálculo 8 bit, tiene lugar la transformación recogida en la tabla 1.4.

Tabla 1.4 Transformación de coeficientes filtro FIR

Coeficientes de entrada	-0.0078	0.0645	0.4433	0.4433	0.0645	-0.0078
Coeficientes transformados	-1	16	113	113	16	-1

De esta manera, en las operaciones a realizar en el Bloque Multiplicador solo intervienen coeficientes enteros.

Salida de Datos.

El generador on-line de Spiral proporciona el resultado presentado en la tabla 1.5.

En el Anexo A-2 se recoge, a modo de ejemplo, el Código Verilog correspondiente al resultado generado tras procesar el ejemplo de Filtro FIR cuyos parámetros de diseño se indican en la tabla 1.4.

El código Verilog presenta tres secciones diferenciadas:

- Cabecera y Datos de los coeficientes del Filtro FIR seleccionado.
- Datos del Modulo Multiplicador.
- Datos de puertos de comunicación y registros E/S.

Tabla 1.5 Respuesta Spiral Online

Result

Integer A constants: 256
Integer B constants: -1 16 113 113 16 -1
Your filter has been generated with the following parameters:
Bitwidth Required: 0 mantissa + 8 fractional = 8 total
Bitwidth Requested: 32
Module Name: acm_filter
Input Data Net: inData
Output Data Net: outData
Clock line: clk
reset option: -reset reset -reset_edge negedge
filterForm: Type 1
Debug mode - extended area information

./firGen.pl '-1' '16' '113' '113' '16' '-1' –moduleName acm_filter
–fractionalBits 8 –bitWidth 32 –inData inData –inReg –outReg –outData outData
–clk clk –reset reset –reset_edge negedge –filterForm 1
–debug –outFile ../outputs/filter_1344443414.v 2>&1
area estimate = 34434.893601497 lambda^2
[[Download Verilog]]

En la tabla 1.6, se indica la posición de comienzo de cada sección en el código Verilog del ejemplo presentado en el Anexo A-2.

Tabla 1.6 Codigo Verilog Esquematico

Líneas	**Descripción de la tarea**
1-7 8-10	Cabecera del código Datos de los Coeficientes del Filtro FIR introducidos y normalizados a la potencia $2^{\text{fraccional bits}}$
12 13-39 41-58 60-65 67	Descripción del Módulo Multiplicador: Puertos de comunicación del Módulo. Conexiones entre nodos de suma/resta/desplazamientos. Conexiones de salida del Módulo Multiplicador. Superficie estimada del circuito del Módulo Multiplicador (Λ^2)
73 74-84 86-107 109-140 141	Descripción del Filtro FIR: Puertos de comunicación del Módulo. Registros de E/S del Módulo y Comportamiento de los registros. Conexiones Filtro - Módulo Multiplicador y Comportamiento. Superficie estimada del circuito del Filtro completo (Λ^2).

El programa Spiral utiliza el software Multiplier Block Generator [10] para generar el grafo correspondiente, ver figura 1.32 y que responde al siguiente esquema de cálculo.

$$113 = (X << 3 - X) << 4 + X$$
$$16 = X << 4$$
$$-1 = \underline{X}$$

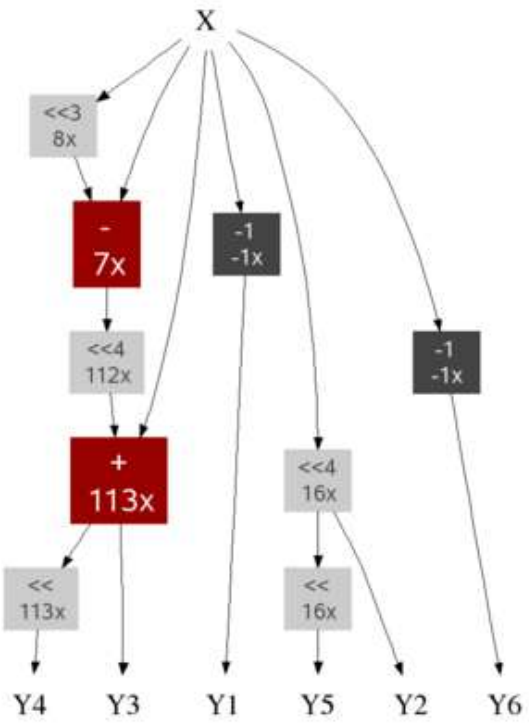


Figura 1.32 Grafo Filtro Paso bajo

Capítulo 2 - Objetivo del presente trabajo.

El objetivo de este Trabajo de Fin de Máster en Ingeniería de Computadores es el diseño de un nuevo tipo de Bloque Multiplicador de Filtro FIR que resuelva la problemática que tiene lugar en los Bloques Multiplicadores estándar actualmente utilizados, y debido a la cual no es posible, en todas las ocasiones, detectar todos los Soft Errors generados en dichos Bloques.

Para clarificar este objetivo, se realizará en primer lugar una descripción pormenorizada del comportamiento de los Bloques Multiplicadores estándar, que pondrá de manifiesto la problemática inherente a su diseño.

A partir de dichos datos se planteará la solución definitiva, el nuevo Bloque Multiplicador diseñado, que imposibilita totalmente la generación de Soft Errors indetectables.

2.1 Análisis del comportamiento de Bloques Multiplicadores estándar.

2.1.1 Descripción de un Bloque Multiplicador estándar de un filtro FIR.

Tras aplicar el algoritmo propuesto por Spiral para el cálculo de un Bloque Multiplicador, se obtiene un grafo que representa la opción óptima de computación del conjunto de operaciones necesarias para conseguir su plena operatividad.

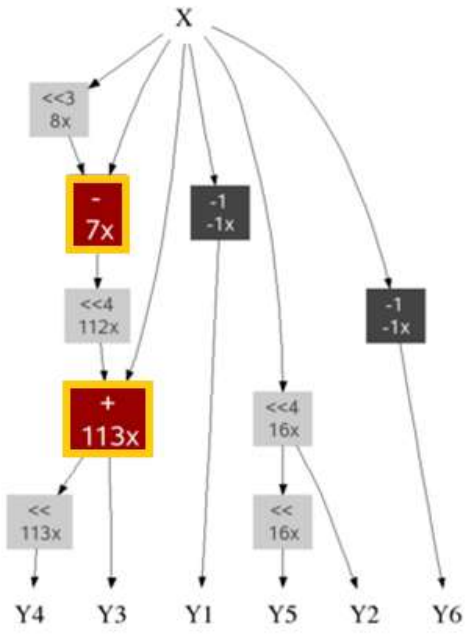


Figura 2.1 Grafo Bloque Multiplicador

Cada una de las cajas con borde amarillo presentes en el grafo de la figura 2.1 representa un sumador/restador de bits.

Para un sumador de dos bits con acarreo, el esquema más simple sería el visualizado en la figura 2.2, en el que se indican las 5 puertas lógicas que lo forman.

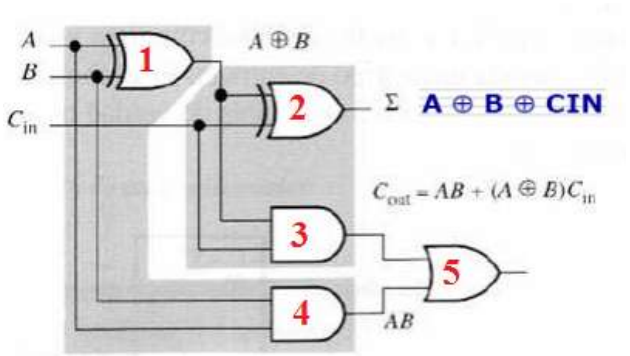


Figura 2.2 Estructura de un sumador de dos bits

Cada puerta lógica está constituida por un conjunto de transistores, hasta 11 en la puerta XOR de la figura 2.3, que son susceptibles de sufrir el impacto de una partícula cargada, experimentando, por tanto, un Soft Error.

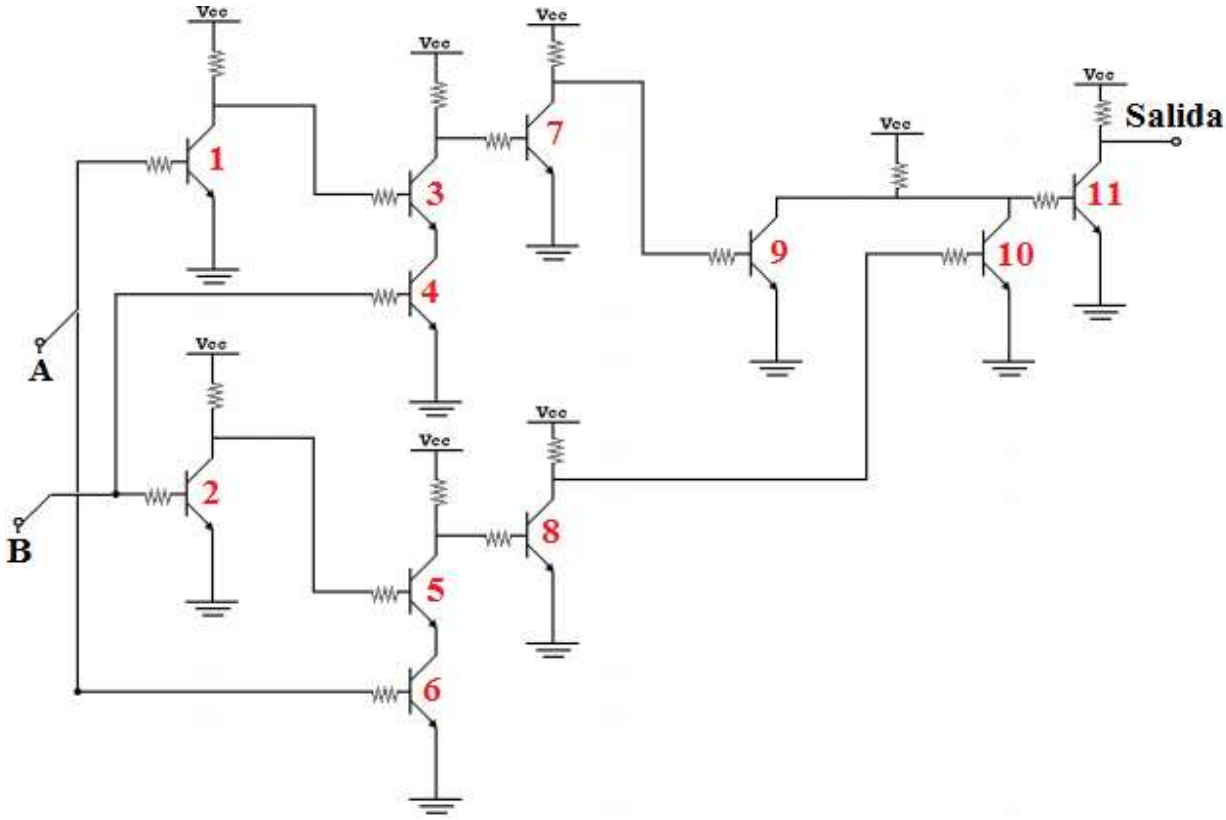


Figura 2.3 Estructura interna de una puerta lógica XOR

Cuando se produce tal situación, la puerta lógica afectada introduce un error en la suma de bits que es arrastrado en la cadena de sumas/restas del Bloque Multiplicador.

Como consecuencia la salida final del Bloque Multiplicador es errónea y el Filtro FIR no operará de forma correcta. Por ello es necesario establecer algún criterio para determinar si se ha producido un Soft Error.

2.1.2 Criterio de detección de Soft Errors en un Bloque Multiplicador estándar.

El primer criterio empleado históricamente para la detección de errores fue la utilización de Redundancia Dual Modular (DMR). Al duplicar el Bloque Multiplicador, si los resultados proporcionados por ambos bloques son diferentes, confirmamos la existencia de Soft Error DUE (Detected Unrecoverable Error). La detección se realiza a costa de utilizar una estructura multiplicadora muy compleja y costosa.

El criterio más sencillo para detectar un suceso Soft Error, se fundamenta en la propia estructura y funcionalidad del Bloque Multiplicador y en la secuencia de acciones que genera el propio Soft Error.

Se expone a continuación:

Propiedad distributiva de un Bloque Multiplicador.

Dado que la suma de los coeficientes de un Bloque Multiplicador es una constante, en ausencia de Soft Error la suma de todas las salidas del bloque debe ser igual al producto de la variable de entrada x, por el sumatorio de los coeficientes que definen dicho bloque. (Figura 2.4).

$$\sum_{k=0}^{n} x\ t_k \ = x \sum_{k=0}^{n} t_k$$

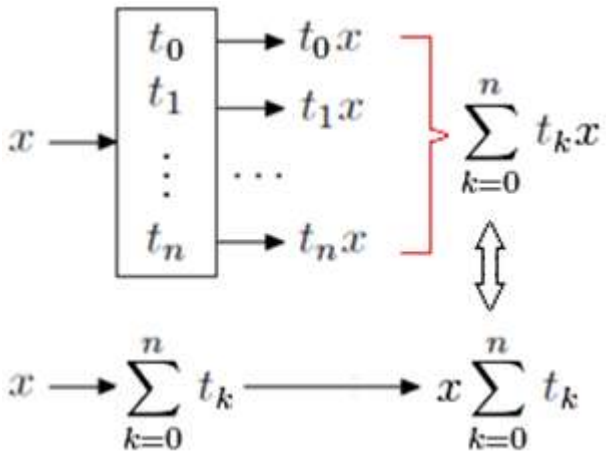


Figura 2.4 Respuesta del Bloque Multiplicador

Propagación del efecto producido por el Soft Error

Una vez producido el Soft Error en uno de los multiplicadores, el error originado se propaga por el resto de multiplicadores, conectados según establece el grafo característico de dicho Filtro.

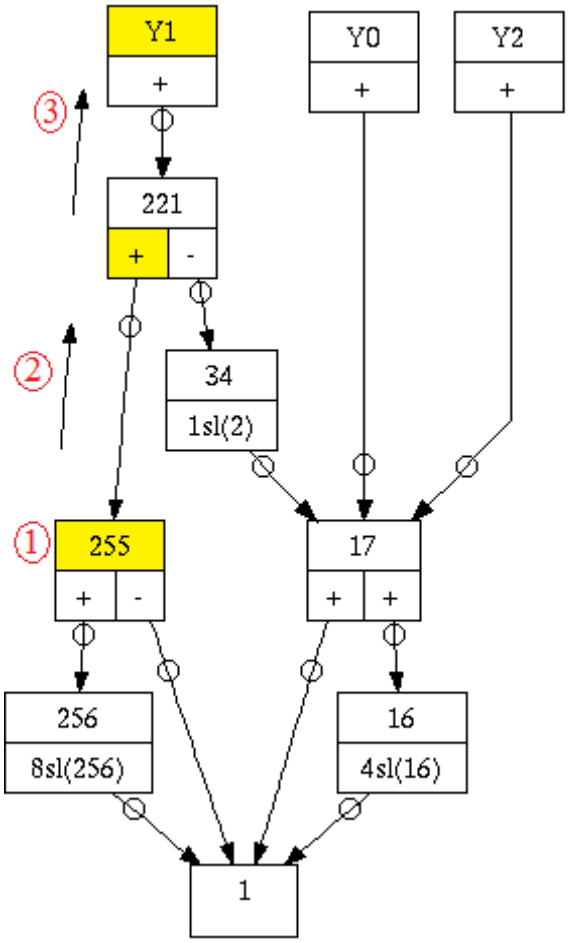


Figura 2.5 Propagación del Soft Error

La propagación, indicada en la figura 2.5 tiene lugar de acuerdo a esta secuencia:

1) Soft Error en multiplicador 255.

2) El error se traslada a la suma en el multiplicador 221.

3) La salida Y1 es errónea.

En estas condiciones, el sumatorio de las salidas no coincide con el producto de la variable de entrada por la suma de los coeficientes del Bloque Multiplicador.

$$\sum_{k=0}^{n} x\ t_k \ - \ x \sum_{k=0}^{n} t_k \ = \ \pm error$$

Por consiguiente, a primera vista parece que implementando en el Bloque Multiplicador un sumador de todas las salidas podríamos detectar la aparición de un Soft Error.

Este criterio es necesario pero no es suficiente, ya que en ocasiones, como se indica en el caso reflejado en el grafo de la figura 2.6, una vez producido el error, puede compensarse induciéndonos a engaño.

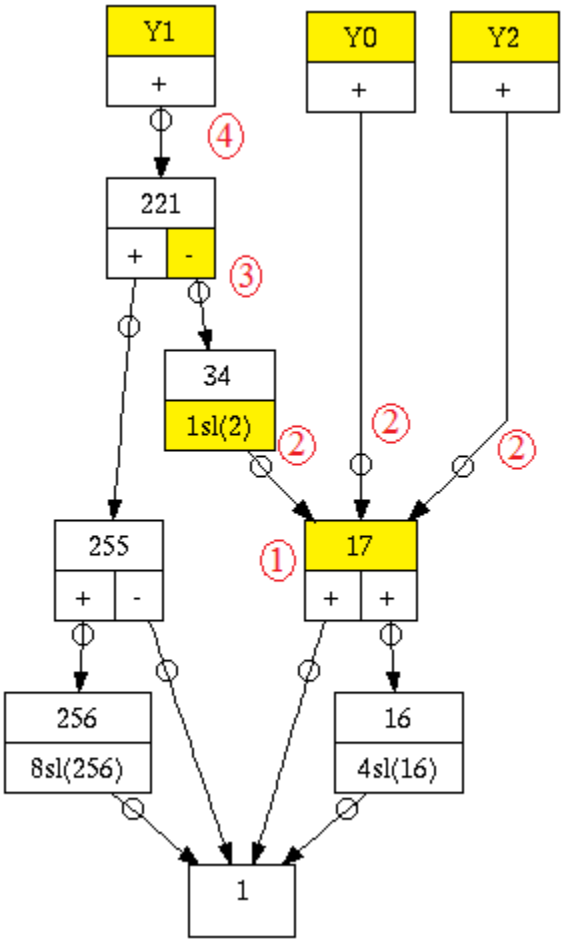


Figura 2.6 Soft Error Indetectable

En este ejemplo, la secuencia de propagación es:

1) Soft Error en multiplicador 17.
2) El error se traslada, afectando *positivamente* a la suma en el multiplicador 34, y a las salidas Y0 e Y2, que generan resultados erróneos.
3) El error afecta, *negativamente* a la suma en el multiplicador 221 multiplicado por un factor de 2.
4) La salida Y1 es también errónea.

En estas condiciones, el sumatorio de las salidas **aparentemente** coincide con el producto de la variable de entrada por la suma de los coeficientes del Bloque Multiplicador.

$$\sum_{k=0}^{n} x\ t_k\ -\ x \sum_{k=0}^{n} t_k\ =\ 0$$

Pero en realidad el error generado en la salida Y1, -2*error, se compensa con los errores generados en las salidas Y0 e Y2, que en ambas ocasiones es +1*error.

$$\sum_{k=0}^{n} x\ t_k\ -\ x \sum_{k=0}^{n} t_k\ =\ error\ -2 * error + error = 0$$

Como puede comprobarse, cuando existe compartición de nodos en el Bloque Multiplicador, aunque la suma de todas las salidas del bloque sea igual al producto de la variable de entrada por el sumatorio de los coeficientes que definen dicho bloque, no se puede confirmar plenamente si se ha producido o no un Soft Error.

2.2 Objetivo de este Proyecto.

El objetivo del presente proyecto titulado "Detección de Soft Errors en Bloques Multiplicadores de Filtros FIR" es la resolución de la incapacidad del Método Spiral para detectar la aparición de Soft Errors en los Bloques Multiplicadores estándar actualmente utilizados,

Para confirmar la consecución del objetivo se contempla la realización de estas tareas:

- ***Propuesta e implantación del Método de Detección de Errores en Bloques Multiplicadores de Filtro FIR sin posibilidad de experimentar Soft Errors indetectables.***

Se debe presentar un diseño original de Bloques Multiplicadores, que ponga fácilmente de manifiesto la aparición de cualquier Soft Error durante la realización de las operaciones aritméticas generadas en su interior. Se describirá las características diferenciales con el diseño estándar.

Se debe implementar el software necesario para diseñar de forma automática los nuevos Bloques Multiplicadores provistos de Registro Check y nodos sin compensación de errores, que imposibiliten la generación de de Soft Errors indetectables.

- ***Evaluación del Método de Detección propuesto y comparativa con la técnica DMR.***

Dado que la superficie de implementación de cualquier circuito ASIC influye de manera decisiva en el coste de adquisición, se debe determinar la superioridad del nuevo diseño de Filtro FIR sin compensación de errores mediante comparación del área del circuito lógico necesario para su fabricación con los correspondientes valores de dos opciones existentes para la detección de Soft Errors: el Método Spiral estándar y la técnica de Redundancia Dual Modular.

- ***Comprobación mediante simulación ModelSim de la eficacia de la técnica de Replicación Parcial en la detección de Soft Errors.***

Para seguir las indicaciones del procedimiento estándar establecido por la IEEE, se debe confirmar la detección total de Soft Errors en la nueva estructura del Bloque Multiplicador con Registro Check y nodos sin compensación de errores propuesto, mediante simulación digital, utilizando para ello la herramienta ModelSim.

Capítulo 3 - Propuesta e Implementación del Método DetectError.

3.1 Bloque Multiplicador DetectError: Descripción.

El nuevo Bloque Multiplicador de Filtros FIR que se propone en este proyecto, llamado BM DetectError, consta de dos elementos diferenciados:

- Registro Check.
- Bloque Multiplicador sin compensación de errores.

En la figura 3.1, se presenta un ejemplo ilustrativo de BM DetectError de un Filtro FIR Paso bajo de orden 4.

A continuación se describen las características diferenciales de cada elemento.

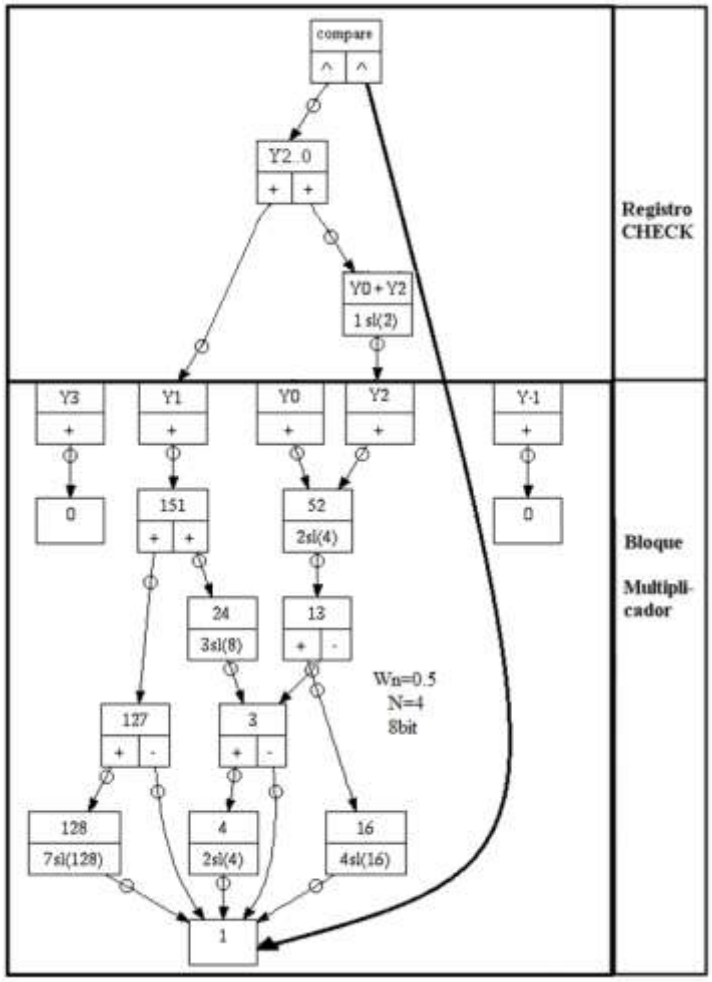


Figura 3.1 Bloque Multiplicador DetectError para Filtro FIR Paso bajo de Orden 4

Bloque Multiplicador sin compensación de errores.

Teniendo en cuenta todo lo indicado en el punto 2.1, el diseño de este nuevo Bloque Multiplicador, debe impedir la compensación de los errores producidos en nodos compartidos, haciendo desaparecer la posibilidad de existir Soft Errors indetectables.

En todo Bloque Multiplicador existen tres tipos de nodos intermedios:

- Tipo Desplazamiento:

  Realiza las multiplicaciones potencia de 2, con desplazamientos de bits.

- Tipo Suma:

  Realiza la suma de sus operandos.

- Tipo Resta:

  Realiza la sustracción de sus operandos.

De acuerdo con la jerarquía funcional del Bloque Multiplicador, cada una de las salidas se denomina Nodo Padre. Cada Nodo Padre se relaciona con uno o más Nodos Hijos. Esta relación queda reflejada mediante la inclusión de un vector, cuya dirección une los Nodos implicados y cuyo sentido apunta al Nodo Hijo, tal y como se indica en la Figura 3.2

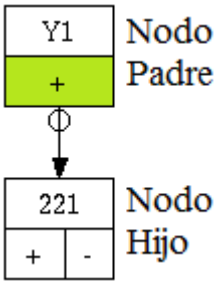


Figura 3.2 Nodos Padre e Hijo

En el caso de nodos Suma o nodos Resta el Nodo Hijo se compone de: Nodo Hijo Izquierdo (NHI) y Nodo Hijo Derecho (NHD), llamados así por estar situados, respectivamente, a la izquierda y derecha de la representación del Nodo en el grafo.

Lógicamente, a su vez, cada uno de los Nodos Hijo es un Nodo Padre con respecto a los nodos con los que está conectado y que se encuentran más alejados de la salida, hasta llegar finalmente al Nodo de entrada del Bloque Multiplicador.

Para cuantificar la propagación de un Soft Error producido en un nodo, en su camino hacia la salida correspondiente, se define la propiedad denominada Característica, representada por la letra C mayúscula.

Cada Nodo Padre asigna a sus nodos Hijos su propia Característica C específica. Su valor se determina haciendo uso de la fórmula de cálculo de propagación del error que se indica en la Tabla 3.1.

El proceso de cálculo comienza por tanto en cada una de las salidas, primeros Nodos Padres, a los que se asigna Característica unidad. El proceso se completa cuando se revisan todos los Nodos intermedios hasta alcanzar el Nodo de entrada al Bloque Multiplicador. Este orden es el contrario que recorrerá la perturbación que se origina en cualquier Nodo afectado por un Soft Error y que se transmite a su Nodo Padre superior, hasta llegar finalmente a una o varias Salidas del Bloque.

Tabla 3.1 Formula de cálculo de la Característica trasmitida

Tipo de nodo intermedio Padre	**Característica transmitida al Hijo**	
Desplazamiento (con valor D)	$C_{hijo}= 2^{D} *C_{padre}$	
Suma	$C_{NHI}= +1*C_{padre}$	$C_{NHD}= +1*C_{padre}$
Resta	$C_{NHI}= +1*C_{padre}$	$C_{NHD}= -1*C_{padre}$

Los Nodos intermedios generados por Spiral pueden presentar Característica positiva, negativa o nula. Solo tiene lugar la compensación de errores en los nodos con Característica nula.

Cuando en un Nodo intermedio determinado, conectado a una o varias salidas, con suma no nula de Característica, se produce un Soft Error, como el error no se compensa en su paso por los siguientes Nodos Padre, todas las salidas implicadas del Bloque Multiplicador son erróneas y, por tanto, su suma no es igual al producto de la entrada por el sumatorio de coeficientes.

Sin embargo, cuando ese nodo posee Característica nula, cada una de las salidas implicadas es errónea aunque aparentemente se cumpla la propiedad distributiva del Bloque y por tanto no se detecta, como tal, el Soft Error que ha ocurrido. Lo que acontece realmente es que la suma de salidas se ha compensado, debido a los errores producidos.

Para impedir esta situación, ninguno de los Nodos del nuevo Bloque Multiplicador propuesto en este Proyecto podrá sufrir compensación de errores.

Para asegurar dicha condición se debe determinar la Característica transmitida por todos y cada uno de los Nodos del Bloque Multiplicador, realizando posteriormente la Replicación de Nodos solo en los casos en que sea necesario.

Replicación de Nodos.

Con este término se describe el proceso consistente en descomponer aquel nodo que pueda producir compensación de errores, en otros dos nodos, cada uno de los cuales presente suma total de Característica distinta de cero, y que en conjunto proporcionen al Bloque Multiplicador el mismo resultado aritmético del nodo original.

De esta forma, ante un Soft Error, no se producirá compensación de errores y por tanto dicho suceso quedará puesto claramente de manifiesto.

Veamos un ejemplo que clarifique el proceso de Replicación de Nodos.

Para ello utilizaremos el grafo correspondiente al Filtro FIR descrito en la figura 2.5, cuya lista completa de Características se representa en la figura 3.3.

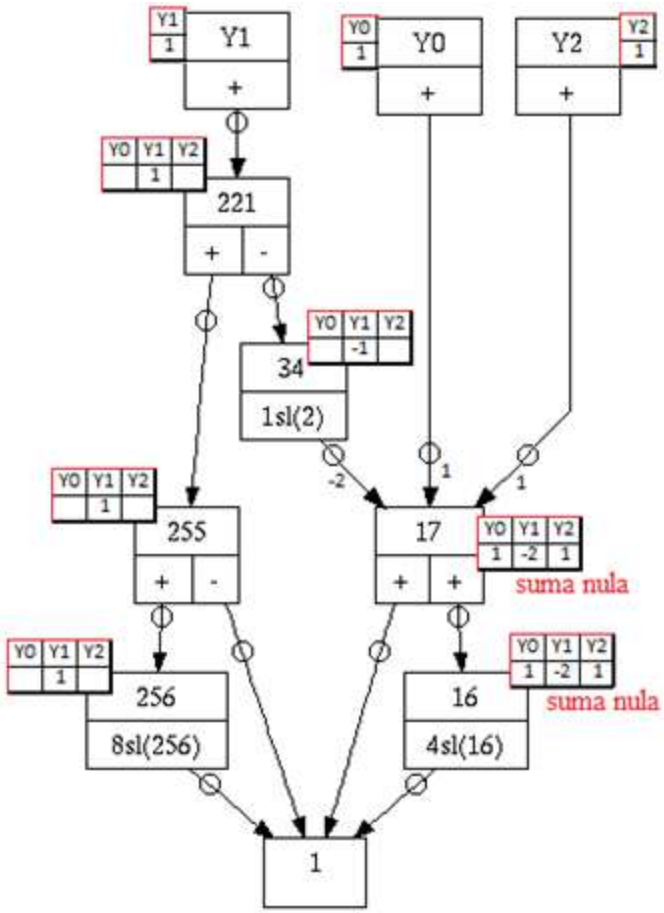


Figura 3.3 Grafo con Características

Como puede comprobarse, los nodos 16 y 17 son los únicos que presentan suma Característica nula.

Dado que el nodo 16 es un Nodo Desplazamiento no es necesario replicarlo por estar construido simplemente a base de conexiones, es decir, sin puertas lógicas que puedan sufrir Soft Errors.

En la figura 3.4 se recoge el proceso de Replicación de Nodos realizado al nodo 17.

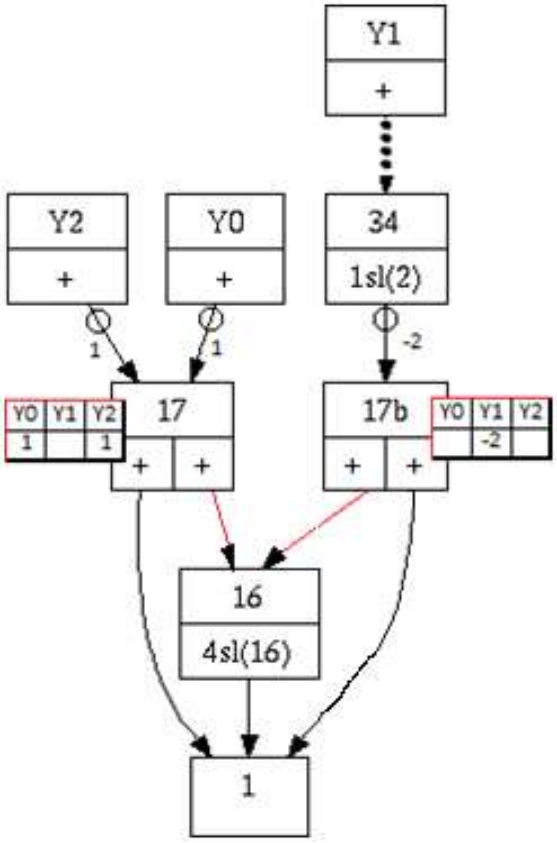


Figura 3.4 Replicación de nodos

El proceso se lleva a cabo de la siguiente forma:

1) Se genera un nuevo nodo del mismo tipo del nodo original (17) unido con los mismos Nodos Hijo, y se le nombra añadiendo la letra ‘b’ (de bis), al nombre del nodo original: 17b.

2) Se traspasa a la tabla de Característica del nodo replicado, la Característica negativa del nodo afectado original, relacionándolo con el Nodo Padre que conduce a la Salida correspondiente.

La Característica traspasada se elimina del nodo original.

3) Se asigna al nodo replicado el mismo valor de área del nodo original.

Registro Check.

Como puede comprobarse en la figura 3.1, el Registro Check del ejemplo escogido está formado por un nodo desplazamiento (duplicación o suma de las salidas Y0 e Y2), un nodo sumador (salida Y1 con suma de salidas Y0 e Y2), y un Comparador (entrada con suma de salidas total).

Para cualquier otro Bloque Multiplicador, el número de sumadores es una unidad inferior al número de salidas diferentes no nulas.

La detección de Soft Errors tiene lugar en el Comparador mediante un proceso aritmético, en el que se compara la suma de las salidas del Bloque Multiplicador con el producto del valor de entrada por el sumatorio de los coeficientes característicos del Filtro.

Tal y como se indicó en el punto 2.1.2, en ausencia de Soft Error:

$$\sum_{k=0}^{n} x\ t_k\ = x \sum_{k=0}^{n} t_k\ = x * Constante$$

Si consiguiésemos diseñar el filtro de forma que la Suma de los coeficientes fuese la unidad, la detección de Soft Errors sería muy sencilla, ya que

$$\sum_{k=0}^{n} x\ \ t_k\ = x$$

Y solo sería necesario comparar la suma de las salidas del Bloque Multiplicador con el valor de entrada, para discernir si el Soft Error se ha producido o no.

Esta es la estrategia de diseño del Registro Check presentado en este trabajo: Acondicionar, cuando sea necesario, los coeficientes que representan al Filtro para conseguir que su suma sea la unidad y tan solo sea necesario comparar la suma de salidas del Bloque Multiplicador con la señal de entrada.

En estas condiciones podemos asegurar que el Bloque Multiplicador DetectError no tiene posibilidad de experimentar Soft Errors indetectables.

3.2 Software para diseño de Bloques Multiplicadores sin compensación de errores.

Para conseguir el diseño de Bloques Multiplicadores sin compensación de errores, se ha construido la aplicación **DetectError**.

El código fuente de la aplicación **DetectError** se encuentra en el anexo A-3.

En la figura 3.5, se indica el diagrama de flujo del proceso de construcción del grafo que representa un Bloque Multiplicador BM DetectError diseñado con esta aplicación

A continuación se describe cada una de las cuatro etapas que deben ser procesadas para completar el diseño del Bloque Multiplicador sin compensación de errores:

Lectura de Datos del archivo Verilog construido con el generador online Spiral.

El software toma como punto de partida el archivo Verilog que describe un Filtro FIR generado online con el programa Spiral, y cuyo esquema típico de representación de componentes y comportamientos, se indica en la tabla 1.6.

Tras abrir dicho archivo, localiza la posición de almacenamiento de los datos del Bloque Multiplicador: salidas y nodos intermedios, procediendo a extraer la información correspondiente a los operandos que intervienen.

De esta manera, para cada línea de Verilog, se creará un nuevo nodo (nodo intermedio) con sus características funcionales. Una vez procesados los datos, el nodo se añadirá a la lista de nodos del nuevo grafo a crear.

Así, para la línea: **“`assign w16 = w1 << 4;`”**, se crea el siguiente nodo:

- nombre “16”.
- tipo “desplazamiento”, valor “4”.
- operando “w1”.

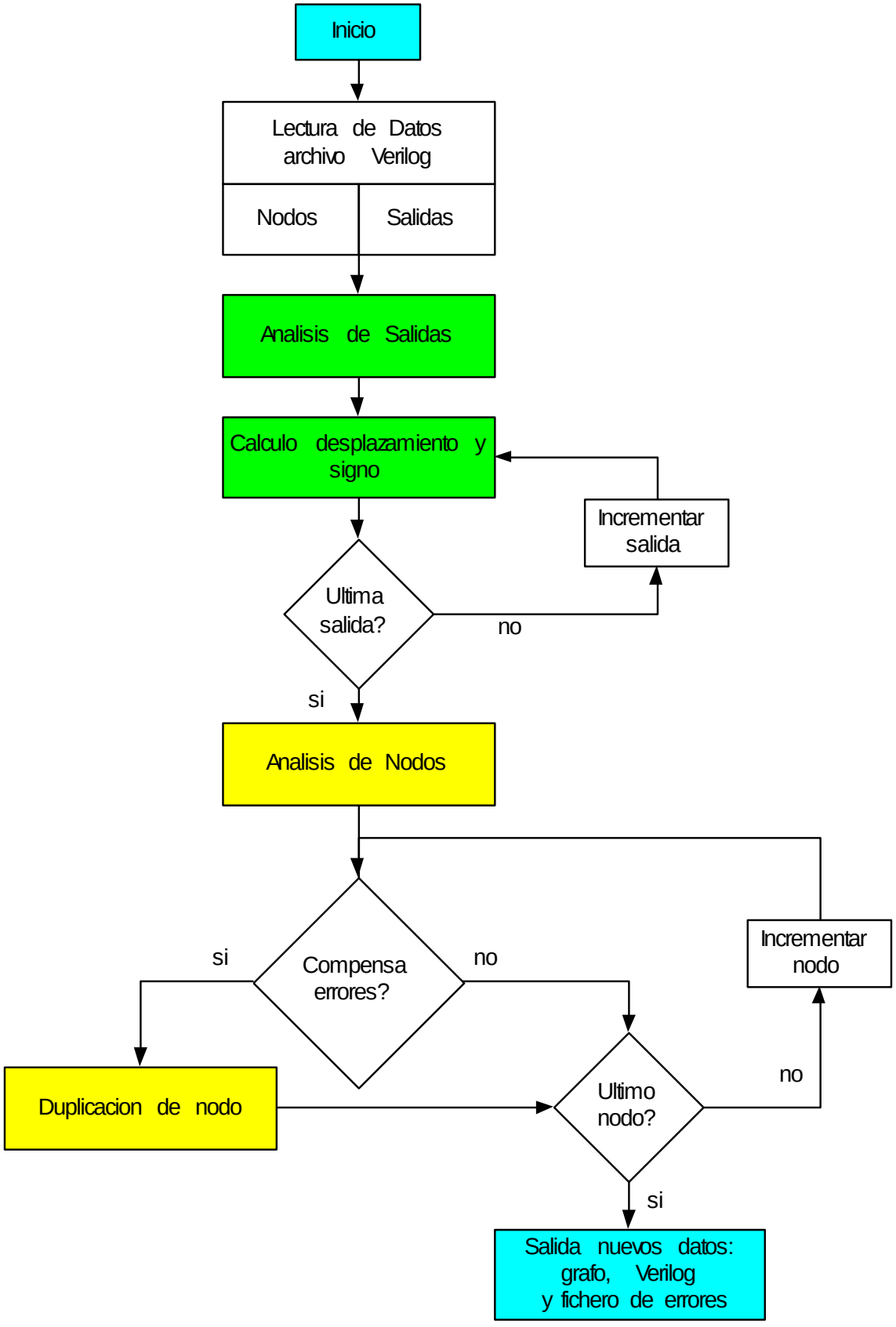


Figura 3.5 Diagrama de flujo calculo grafo sin compensación de error

A continuación procede a localizar y extraer del archivo Verilog original, la información correspondiente a las salidas del Bloque Multiplicador.

Para cada línea de Verilog, se creará un nuevo nodo de salida con su Característica funcional. Una vez procesados los datos, el nodo se añadirá a la lista de nodos de salida del grafo multiplicador.

Así, para la línea: "**`assign Y[0] = w5_[39:8];`**", se crea el siguiente nodo de salida:

- nombre "0".
- tipo "salida", valor "-1".
- operando "w5".

La aplicación cierra el archivo Verilog y pasa el control al siguiente bloque de operación.

Análisis de las Salidas del Bloque Multiplicador.

La aplicación procede a analizar el grafo, cuantificando el efecto que ejercen los nodos intermedios sobre cada uno de los nodos de salida del Bloque Multiplicador.

El análisis se efectúa según la cascada de dependencia, relacionando en cada momento dos nodos. El nodo más próximo a la salida o Nodo Padre y su Nodo Hijo.

Una vez que se han analizado todos los nodos de salida, la aplicación dispone de la lista completa de Características, es decir, la respuesta que el Soft Error generado en cada uno de los nodos aporta a cada una de las salidas.

Con esta información se lleva a cabo la siguiente etapa del programa.

Análisis del comportamiento de los Nodos frente a la compensación de errores.

En esta etapa la aplicación procede a analizar la lista completa de nodos, comprobando en cada caso si se produce o no suma Característica nula.

Tras completar el proceso de Replicación de Nodos para todos los nodos en los que se produce compensación de errores, el nuevo Bloque Multiplicador obtenido ya no puede generar Soft Error indetectables.

Creación del código DOT que visualiza el grafo del Bloque Multiplicador diseñado.

Todos los grafos presentados hasta el momento, se han construido mediante el concurso del editor grafico para el sistema X Window, conocido como **dotty** [11]. Este potente editor gráfico utiliza el lenguaje DOT [12] y se apoya en la funcionalidad del editor Lefty [13].

La aplicación **DectectError**, una vez que ha asignado los distintos nodos que constituyen el Bloque Multiplicador y conoce sus Características y comportamientos, procede a generar el código dotty.

En la Tabla 3.2 se presenta un ejemplo ilustrativo de código DOT.

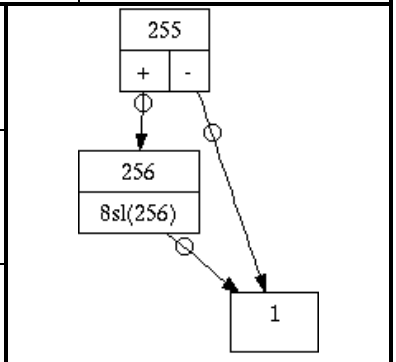
Tabla 3.2 Ejemplo de código DOT.

	Nodo		
Nombre	255	256	1
Tipo	Resta	Desplazamiento	Terminal
operandos	w256, w1		X
Valor		8	

```
Node0x255[shape=record,label="{255|{<s0>+|<s1>-}}"];
Node0x255:s0 -> Node0x256;
Node0x255:s1 -> Node0x1;
```

```
Node0x256[shape=record,label="{256|{<s0>8sl(256)}}"];
Node0x256:s0 -> Node0x1;
```

```
Node0x1[shape=record,label="{1}"];
```



La **sintaxis** de este lenguaje es la siguiente:

Definir nodo: **nombre**[shape=record,label="{**texto**}"];

Subdividir texto de un nodo: **textoNodo**|{<s0>**textoAbajoIzda**|<s1> **textoAbajoDcha** }

Definir una arista con dirección: **origen** -- **destino**;

Definir una arista con dirección y sentido: **origen** -> **destino**;

Para la **generación** del código dotty del grafo se ha implementado la función **toCFG()**.

La función es ejecutada para todos los nodos. Teniendo en cuenta sus atributos (nombre, tipo, operandos, valor), construye la cadena de texto en lenguaje DOT que identifica y define dicho nodo dentro del grafo.

Con el código generado, el programa dotty **visualiza** en pantalla el grafo correspondiente al Bloque Multiplicador diseñado. Ver figura 3.6.

En el anexo A-5 se presenta el código DOT generado para el filtro ejemplo utilizado a lo largo de los anteriores apartados como aclaración del funcionamiento del programa.

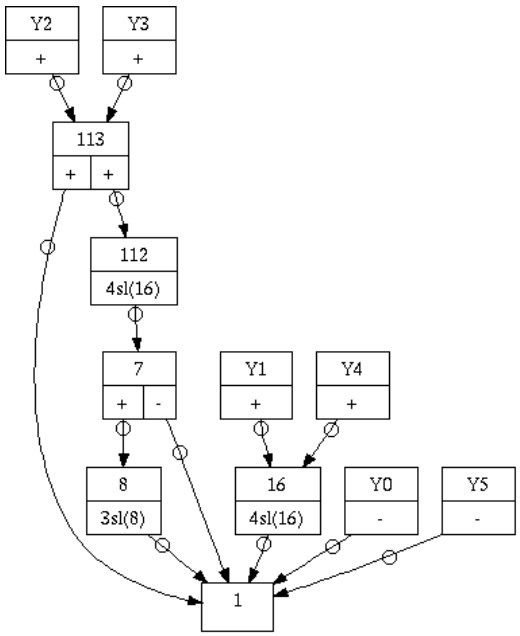


Figura 3.6 Grafo obtenido

Además el programa genera diversos datos auxiliares, que permiten visualizar el correcto diseño del bloque. Entre ellos se encuentran:

- Lectura de datos Filtro FIR.
- Grafo Bloque Multiplicador Spiral.
- Análisis del comportamiento de nodos frente a compensación de error.
- Área utilizada por los nodos.
- Grafo Bloque Multiplicador sin compensación de error.
- Área necesaria para implementar los nodos del Bloque Multiplicador.
- Área del Registro Check.
- InfoSize.

Creación del nuevo archivo Verilog para Bloques Multiplicadores sin compensación de errores.

La aplicación guarda toda la información correspondiente al nuevo Filtro FIR que incorpora un Bloque Multiplicador sin compensación de errores, utilizando el mismo formato Verilog de partida.

En el Anexo A-4 se recoge el archivo Verilog generado por la aplicación **DectectError**.

A continuación se exponen las diferencias con el archivo original:

1) Cabecera de archivo:

   Se introduce como novedad el texto que indica la autoría del programa DectectError, manteniendo la información referente al generador online y los datos de los Coeficientes del Filtro FIR utilizados.

2) Descripción del Módulo Multiplicador:

   a. Puertos de comunicación del Módulo Multiplicador.
      - Se añade como primer puerto, el denominado Puerto de salida Check.

      Este puerto monitoriza el funcionamiento del Bloque Multiplicador ante un Soft Error. Como por diseño el bloque no tiene compensación de errores, mientras presente un valor nulo se confirma la inexistencia de Soft Error.

      - El resto de puertos se presentan en la misma secuencia que en el archivo original.

   b. Conexiones entre nodos de suma/resta/desplazamientos.
      - A la lista de señales originales se añaden las correspondientes a la conexión hacia el nodo Check, y las que parten de cada uno de los nodos replicados.
      - Se incorpora la asignación que describe las nuevas replicas de nodos.

   c. Conexiones de salida del Módulo Multiplicador
      - Se añade la expresión de cálculo de salida del Puerto Check.

3) Descripción del Filtro FIR:

   Se completa la descripción del filtro con la inclusión de las instrucciones necesarias para la revisión de su comportamiento en un simulador digital.

Creación de un archivo de errores para simulación digital con ModelSim.

Aprovechando toda la información disponible acerca del número de nodos, registros y salidas, así como el ancho de bit resultante de la precisión elegida para el diseño del Bloque Multiplicador, la aplicación DetectError genera un fichero texto secuencial en el que se guardan pares de líneas con la siguiente información:

*) Valor de entrada al Bloque Multiplicador.

*) Entrada con Soft Error o sin incidencia.

Cuando no se fuerza la existencia de error, la línea está en blanco.

Cuando se fuerza dicho error, se indica el nodo y bit afectado, utilizando el mismo formato de Spiral.

Para asegurar un número suficiente de errores que cubra todos los bits de todos los nodos presentes en el Bloque Multiplicador y que el proceso no tenga intervención del usuario, se establecen los siguientes parámetros:

*) Número de Errores por bit para todos los nodos: 10.

*) Número de Errores sobre el total de entradas presentes en el fichero: 80%

*) Elección de nodos y bit: Totalmente aleatoria.

Capítulo 4 - Evaluación del Método y comparativa con DMR.

La respuesta de un Filtro FIR real depende en gran manera del orden del filtro utilizado.

Cuanto mayor sea el orden seleccionado, más se aproximará la respuesta al comportamiento que presentaría un Filtro FIR ideal que procesase la misma secuencia de datos a filtrar, pero, en contrapartida, cuanto mayor sea el orden del Filtro seleccionado, tanto mayor será la carga computacional necesaria para llevar a cabo los cálculos aritméticos en el Bloque Multiplicador y en los registros de acumulación del propio filtro.

Además, es necesario implementar un circuito adicional que notifique la aparición del Soft Error, el Registro Check, cuando se utiliza el Método DetectError o implementar X copias del bloque de operación y sus circuitos adicionales cuando se utiliza la técnica de Replicación Múltiple Modular (XMR).

Es decir, cuanto más complejo sea el modelo matemático que representa al Filtro FIR, más compleja será la estructura lógica que necesita ser implantada físicamente en el circuito electrónico que lo define. Ello dará lugar a un circuito mucho más complicado de construir y por tanto con un coste superior de fabricación.

En este capítulo se procede a cuantificar la influencia que ejerce sobre el tamaño y el coste de diseño de circuitos ASIC de Filtros FIR con detección de Soft Errors, todos estos factores, orden del filtro, grado de precisión de cálculo y modo de detección de errores, seleccionando además dos tipos distintos de filtrado digital: Filtrado Paso bajo y Filtrado Paso alto.

4.1 Calculo del área necesaria para la construcción del Registro Check.

La aplicación **DectectError**, procede a utilizar y calcular las siguientes variables:

4.1.1 Adecuación de los Coeficientes del Bloque Multiplicador.

En el caso de Filtros FIR Paso bajo, no es necesario realizar ningún cambio en los valores proporcionados por MATLAB, ya que directamente la suma de coeficientes tiene un valor unidad. En el anexo A1.1 se recogen los datos correspondientes a Filtros FIR de orden 2 a 16.

En el caso de Filtros FIR Paso alto, como la suma de los coeficientes proporcionados por MATLAB es una constante distinta de la unidad y cuyo valor se aproxima tanto más a cero cuanto

mayor es el orden del Filtro, es necesario modificar dichos Coeficientes originales para obtener unos Coeficientes Finales que puedan ser usados sin problemas para el cálculo del Filtro.

El proceso tiene lugar en dos etapas.

En la primera se obtienen unos Coeficientes Normalizados cuya suma es la unidad (potencia cero de 2) y que por tanto puede ser representada mediante un único bit.

Para ello se divide en cada caso el valor del coeficiente original MATLAB por el valor de la suma total de coeficientes.

Este primer paso es necesario pero no suficiente para asegurar el correcto funcionamiento del programa Spiral ya que en ocasiones los coeficientes normalizados son tan excesivamente grandes que Spiral genera errores por desbordamiento durante el cálculo.

Es en la segunda etapa donde se procede a acotar el valor máximo de los coeficientes.

Para ello se recalcula el valor definitivo de cada uno de los coeficientes, dividiéndolo por la potencia de 2 inmediatamente superior al coeficiente de mayor valor.

Los coeficientes quedan acotados entre -1 y 1.

De esta forma ocupan como máximo un bit más que la precisión escogida en cada caso y en ningún caso, ni para precisión de cálculo 24, los coeficientes definitivos (Coeficientes Finales) hacen superar el tope de 26 bit soportado internamente por Spiral, siendo, por tanto, posible utilizar su generador online.

En el Anexo A-1.2, se recogen los Coeficientes Finales para Filtros FIR Paso alto de orden 2 a 16, obtenidos al aplicar el procedimiento indicado anteriormente.

4.1.2 Calculo del Área necesaria para implementar el Registro Check.

Para realizar la suma de N+1 salidas de un Bloque Multiplicador, siendo N el número de orden del Filtro, es necesario concatenar N sumadores y un Comparador, encargado de validar la no aparición de Soft Errors.

El valor de Área que ocupará un Registro Check, puede calcularse mediante la siguiente expresión:

Área Registro_Check = N x Área de Sumador_Check + Área Comparador_Check

A continuación se estimará cada una de estas áreas.

Estimación del área de un sumador Check.

Spiral asigna a cada uno de los sumadores de un Bloque Multiplicador, un valor de área que se puede calcular en función del número de bits implicados en la operación, según las siguientes expresiones basadas en su librería de hardware **cmulib18**:

Área estimada del Sumador (adderArea)

Ejemplo: w257 =w1 +w256 adderArea(8,39)

adderArea = 69,8544* (39-8+1) – 39,9168 = 2195,4240 λ^2

Área estimada de un Restador (subArea)

Ejemplo: w3 =w4 -w1 subArea(2,33)

subArea = 76,5072 * (33-2+1) – 39,9168 = 2408,3135 λ^2

Área estimada de un Negador (negArea)

Ejemplo w8_ = -1 *w8 negArea(3,34)

negArea = 46,5696 * (34-3+1) – 53,22241 = 1437,0048 λ^2

Teniendo en cuenta que cada uno de los Sumadores del Registro Check es idéntico a cualquier sumador contenido en los registros de acumulación de un Filtro FIR, con la única diferencia de que el número máximo de bits implicados en la operación de suma es 32, podemos calcular su Área, utilizando la misma expresión indicada por Spiral.

En consecuencia:

Área estimada de un Sumador del Registro Check (adderArea_Check)

adderArea_Check = 69,8544* (32) – 39,9168 = 2195,4240 λ^2

Este es el valor que será utilizado para todos los cálculos realizados en este trabajo.

Estimación del área del Comparador Check.

Para el comparador del Registro Check, utilizaremos un valor de área equivalente a la mitad del área de un sumador Check. Para ello se ha seguido el siguiente razonamiento.

En la figura 4.1 se representan las estructuras lógicas de un Comparador de 2*M bit y un sumador de 2 bit, que contienen:

- Comparador: 2 * M puertas XOR y M-1 puertas AND.
- Sumador: 2 puertas XOR, 2 puertas AND y 1 puerta OR.

Es decir:

Puertas Comparador de 2 * M bit = Puertas Sumador de M bit

O su equivalente:

Puertas Sumador de 2 * M bit = 2 * Puertas Comparador de 2* M bit

Teniendo en cuenta las reglas de diseño de circuitos ASIC indicadas en el punto 10.8 del Anexo A-10, el comparador de 2*M bit de un Filtro FIR DMR y el comparador del Registro Check ocuparán una superficie del orden de la mitad del sumador de 2*M bit, cuyo valor se ha estimado anteriormente:

$$\text{Comparador\_Check} = \frac{\text{adderArea \_Check}}{2} = \frac{2195{,}4240\ \lambda 2}{2} = 1097{,}7120\ \lambda^2$$

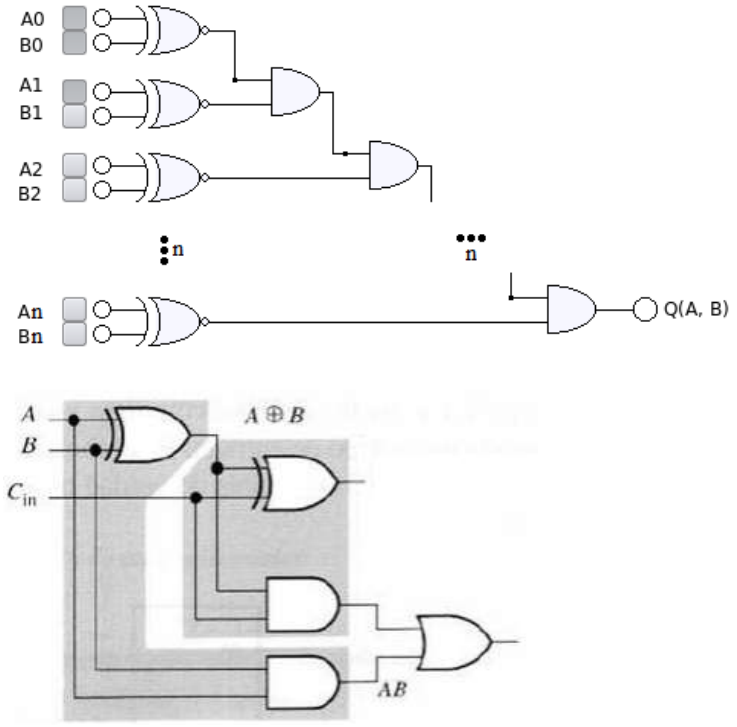


Figura 4.1 Estructura lógica de un comparador de n bits y un sumador de 2 bits.

Estimación del área total del Registro Check.

Teniendo en cuenta la simetría de los Filtros FIR, el Área de un Registro Check DetectError se puede calcular de forma aproximada mediante la expresión:

$$\textbf{Área Registro Check = 2195,4 * [(Nnonuldif -1) + 0,5]}\ \lambda^2.$$

Siendo **Nnonuldif**, el número de Coeficientes no nulos diferentes de salida del Registro Check.

En el ejemplo de la figura 3.1, se comprueba la existencia de dos salidas nulas, Y-1 e Y3. De acuerdo con lo indicado anteriormente, Nnonuldif = 5 - 2 – 1 = 2, y por tanto, para dicha situación, el Área del Registro Check será:

$$\text{Área Registro Check} = 2195{,}4 * [(2 - 1) + 0{,}5]\ \lambda^2 .= 3293{,}1\ \lambda^2$$

4.2 Cálculo de superficie de Filtros FIR Paso bajo con detección de Soft Errors.

En este apartado se procederá al estudio del comportamiento de Filtros FIR, determinando la superficie necesaria para implementar en un circuito ASIC, el Bloque Multiplicador, los registros de acumulación y el Registro Check de un Filtro FIR Paso bajo con Bloque DetectError, que operen en las siguientes condiciones:

- Orden de filtro: 2 a 16.
- Frecuencia normalizada de corte: 0’1 a 0’9 en intervalos de 0’1.
- Precisión del Bloque Multiplicador: 8, 12, 16, 20 y 24 bit.

Además, para confirmar que el Método DetectError presenta ventajas sobre otras estrategias con y sin tratamiento de errores, se determinará también dicha superficie para Filtros FIR Paso bajo que operen con Bloque Multiplicador estándar (Bloque Spiral) y Filtros FIR Paso bajo con Redundancia Dual Modular (Bloque Spiral y tratamiento de errores).

Para conseguir dicha información se han considerado las siguientes etapas:

4.2.1) Obtención de grafos, Área de Bloque Multiplicador y Área total de circuito de Filtros FIR Paso bajo estándar (Bloque Spiral).

4.2.2) Obtención de grafos, Área de Bloque Multiplicador que incluye nodo detector de errores y Área total de circuito de Filtros FIR Paso bajo con nodos replicados (Bloque DetectError).

4.2.3) Obtención del Área del Bloque Multiplicador que incluye comparador para detección de errores y Área total de circuito de Filtros FIR-DMR Paso bajo (Bloque Spiral y tratamiento de errores).

4.2.1 Obtención de grafos, área de Bloque Multiplicador y Área total de circuito de Filtros FIR Paso bajo estándar (Bloque Spiral).

Se ha recopilado el código DOT representativo de todos los Bloques Multiplicadores estándar generados por Spiral para las condiciones indicadas en el punto 4.2, representando el grafo correspondiente con la aplicación dotty.

Se ha utilizado esta aplicación, ya que genera grafos con mucho más detalle que Spiral, tal y como puede comprobarse en la figura 4.2, en donde se compara, a modo de ejemplo, los grafos obtenidos para un Filtro FIR Paso bajo de orden 6, y frecuencia normalizada de corte 0'9.

Además, en los grafos obtenidos con la aplicación dotty, se visualiza claramente el sentido de transferencia de Característica entre Nodos Padre y Nodos Hijo, propiedad que posibilita la obtención de Nodos sin compensación de errores.

En el Anexo A-6 se recogen los grafos correspondientes a los Filtros FIR Paso bajo de orden 2 a 10, con precisión 8 bits considerados en este trabajo.

Dado el gran tamaño de los grafos correspondientes a Filtros FIR Paso bajo con precisión de 16 bits, solo se recogen diez ejemplos correspondientes a Filtros FIR Paso bajo que operan con frecuencia normalizada de corte 0'5.

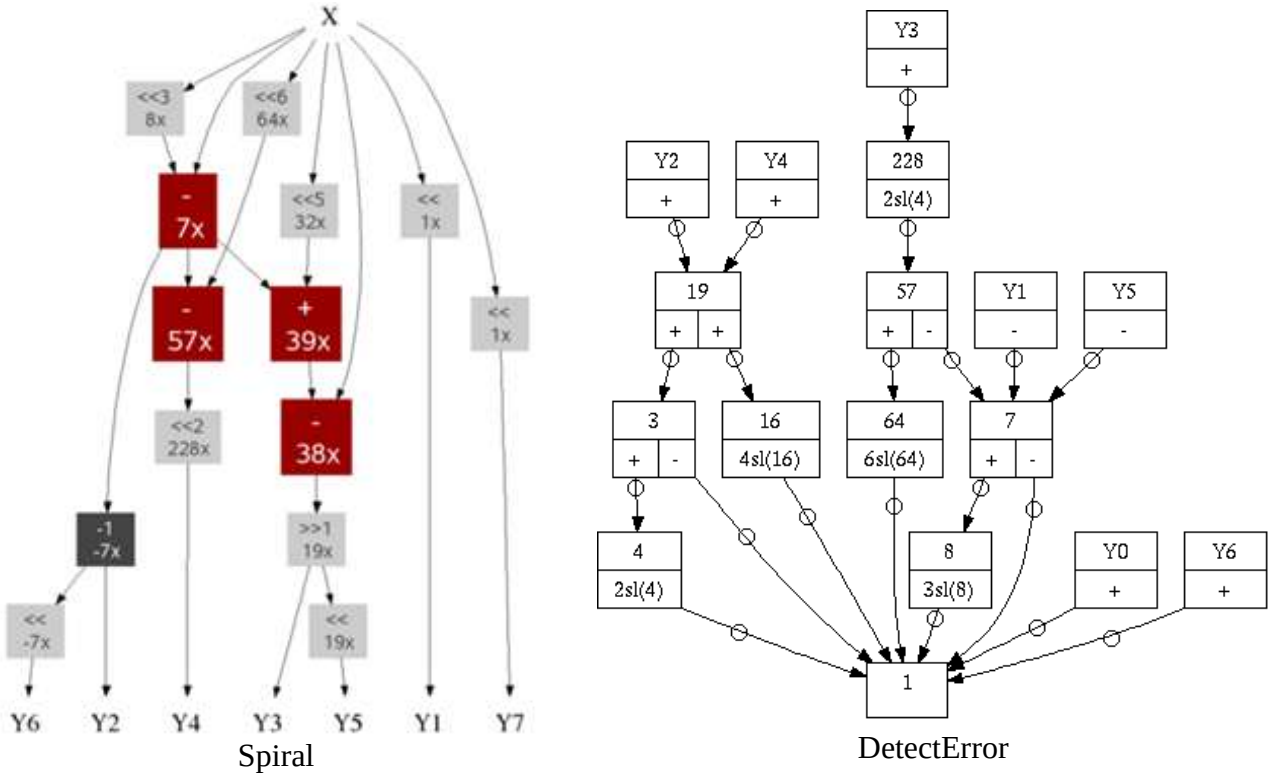


Figura 4.2 Grafos generados por Spiral y por DetectError

Área Bloque Multiplicador y Área total de Filtro FIR Paso bajo (Bloque Spiral).

En las tablas 4.1 a 4.10, se presentan los datos obtenidos mediante Spiral, de la superficie de circuito necesario para implementar los Filtro FIR Paso bajo estudiados en el punto 4.2, indicando por separado la contribución del Bloque Multiplicador y la del área total del circuito lógico. De esta manera es posible determinar el peso relativo del Bloque Multiplicador en el coste computacional total del Filtro FIR Paso bajo.

Como puede observarse en varias de las tablas correspondientes al Área de Bloque Multiplicador, Tablas 4.1, 4.3, 4.5, 4.7 y 4.9, aparecen asignados valores de 0 unidades de superficie.

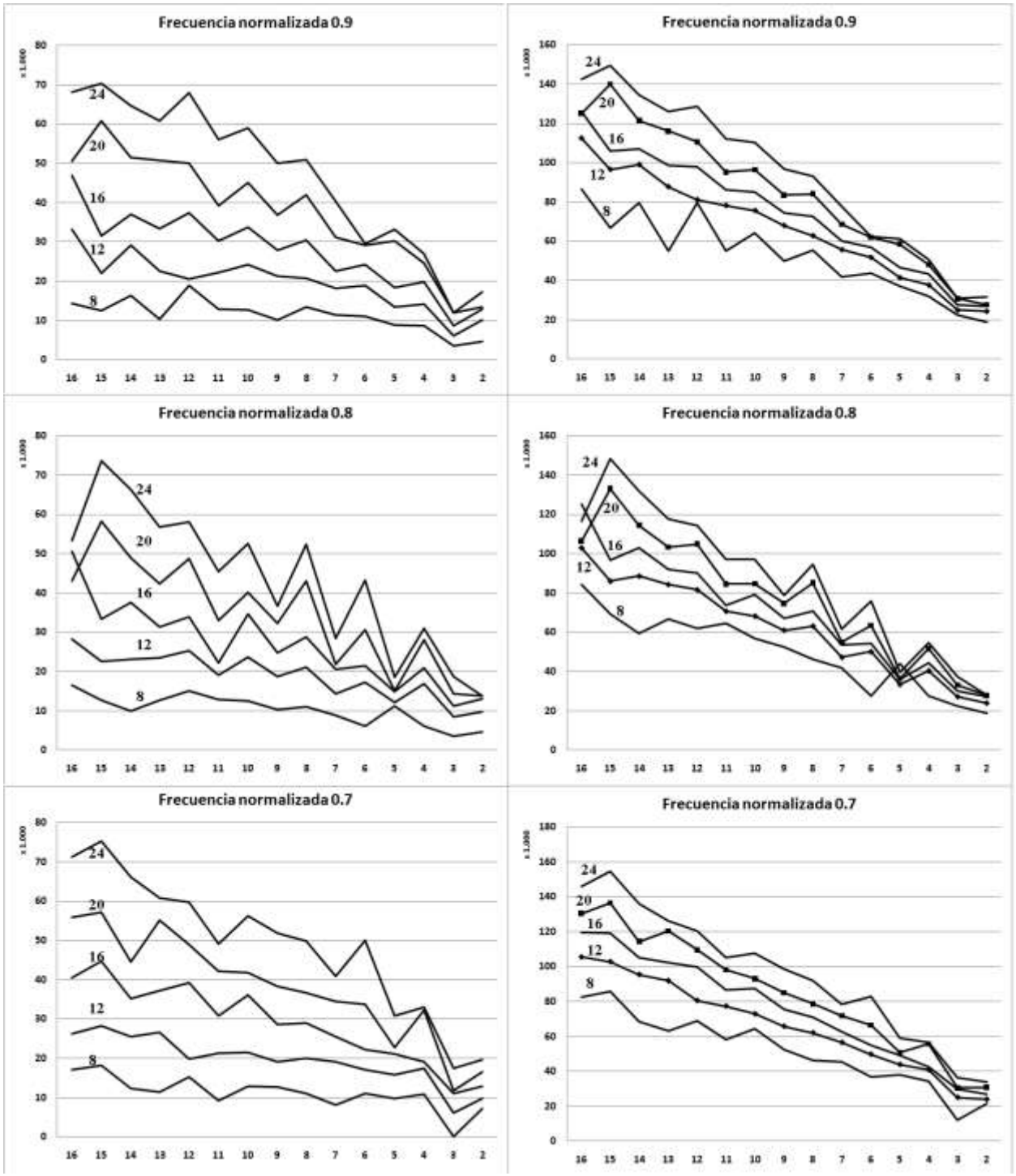
Tomemos como ejemplo en la tabla 4.1, el Filtro FIR Pasabajo de orden 3 y frecuencia normalizada de corte 0’7, con precisión de cálculo 8 bit. El Área del Bloque multiplicador es cero debido a que, al ser los coeficientes de dicho Filtro: -0’00305878, 0’50305878, 0’50305878 y -0’00305878 (0, 128, 128 y 0 en Spiral), las multiplicaciones se convierten en desplazamientos de 0 y 7 bit. En estas condiciones, no es necesaria la existencia de ninguna puerta lógica, solo hay que implementar conexiones y por tanto no hay coste de área alguno.

En las figuras 4.3, 4.4 y 4.5 se representa para cada frecuencia normalizada de corte del filtro, los valores de áreas de Bloque Multiplicador y área total de circuito frente al orden, para Filtros FIR Paso bajo con precisión de cálculo entre 8 y 24 bit.

Como puede comprobarse se confirma el patrón teórico de necesidad de mayores áreas de circuito cuanto mayor es el orden del Filtro FIR y como a igualdad de orden es mayor la superficie cuanto más elevada es la precisión de cálculo, debido a la complejidad creciente de sus Bloques Multiplicadores.

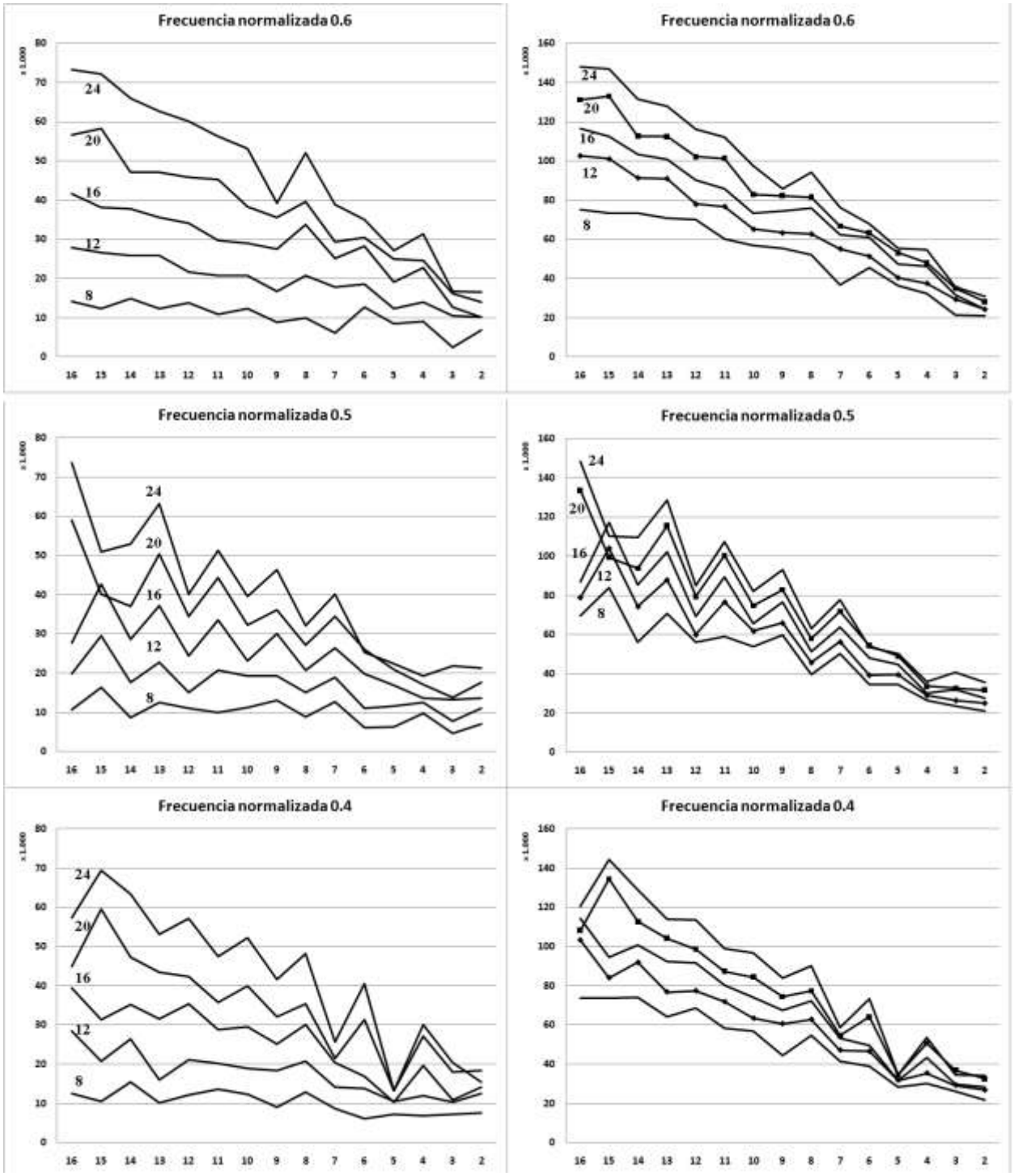
Se observa un comportamiento muy errático para la frecuencia normalizada 0’5. Ello es debido a que, tal y como se observa en las tablas 4.11 (precisión 8 bit) y 4.12 (precisión mayor de 12 bit), varios de los coeficientes proporcionados por MATLAB para dicha frecuencia de corte en filtros de orden par, son inferiores en valor absoluto a la mitad del valor del fractional bit usado y Spiral al transformar dichos coeficientes, acomodándolos a su formato, los considera nulos.

En esas condiciones la salida se conecta a valor cero, sin que sea necesario por tanto un nodo que realice la multiplicación. Ello da lugar a un valor de área inferior al obtenido en aquellos filtros de orden impar, donde no se produce dicha situación.



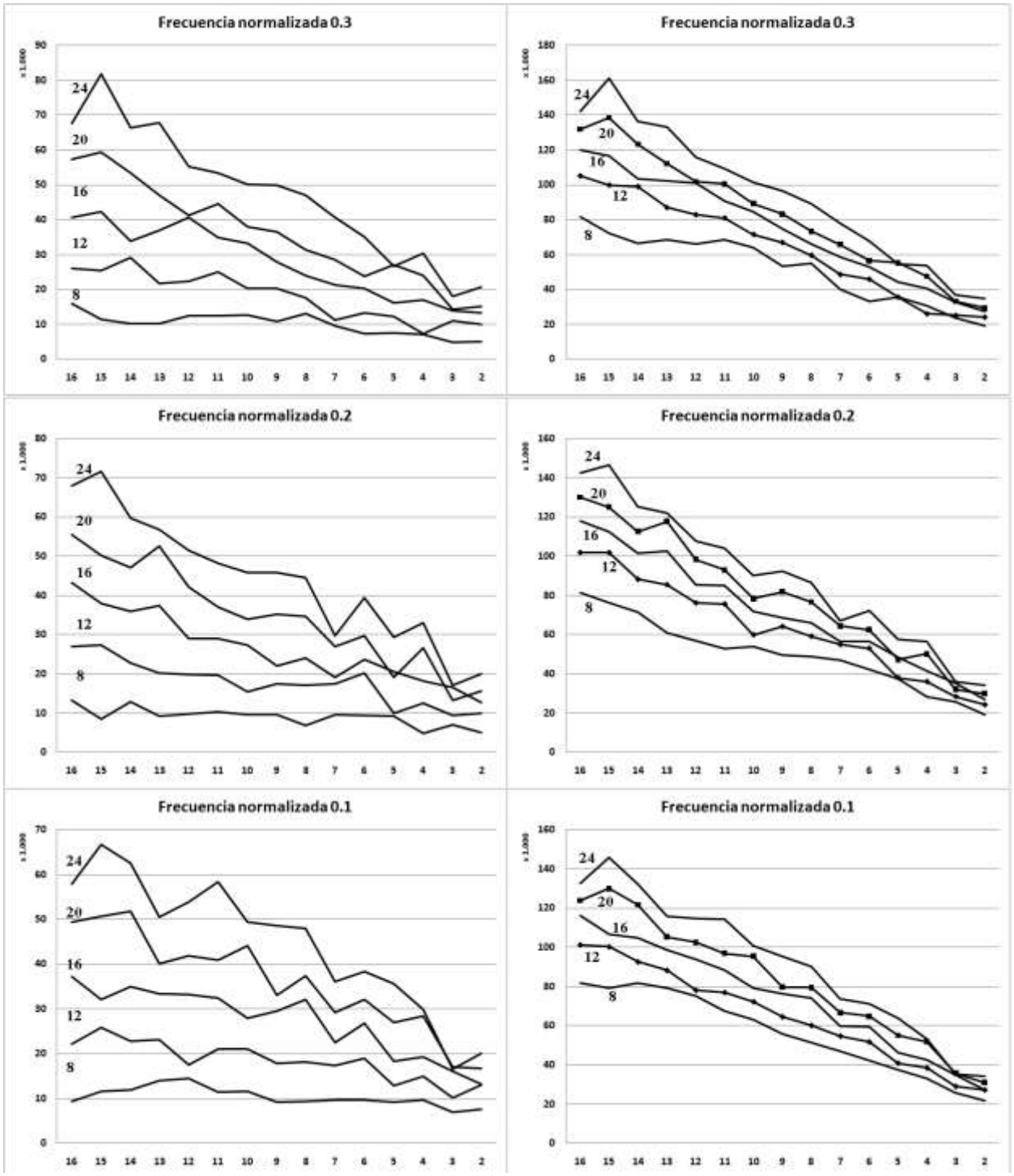
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.3 Área Bloque Multiplicador y Área total circuito FIR Paso bajo (Spiral) (1)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.4 Área Bloque Multiplicador y Área total circuito FIR Paso bajo (Spiral) (2)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.5 Área Bloque Multiplicador y Área total circuito FIR Paso bajo (Spiral) (3)

El porcentaje de área aportado al total del circuito por los Bloques Multiplicadores varía desde una media de aproximadamente el 21% para cálculos con 8 bit de precisión a una media de más del 50% para precisión de 24 bit.

En la figura 4.6 se representa dicha variación según la precisión de cálculo utilizada.

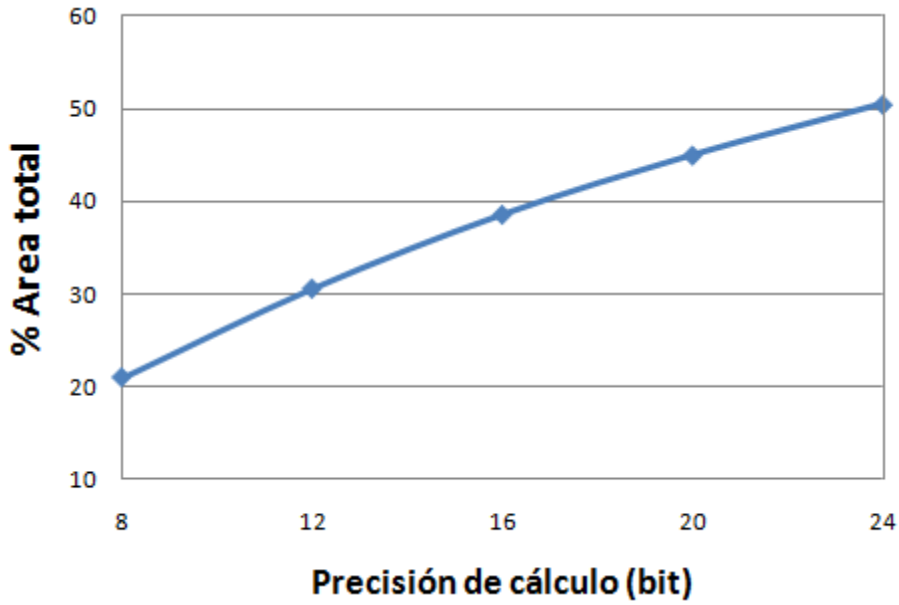


Figura 4.6 Porcentaje Área aportada por Bloque Multiplicador FIR Paso bajo (Spiral)

4.2.2 Obtención de grafos, Área de Bloque Multiplicador y Área total de circuito de Filtros FIR Paso bajo (Bloque DetectError).

La aplicación **DetectError** se ha diseñado con el objeto de construir Bloques Multiplicadores exentos de compensación de errores, con los que se pone claramente de manifiesto la aparición de Soft Errors.

Por ello se ha analizado los datos Verilog de todos los Filtros FIR Paso bajo recogidos en el anexo A-2, construidos desde precisión 8bit a precisión 24bit, comprobando que solo es necesario realizar replicaciones parciales en los casos indicados con la letra "R" en las tablas 4.13 a 4.17. En todas ellas se expresa con el cardinal adecuado el número de nodos que es necesario replicar para conseguir eliminar la compensación de errores.

En el Anexo A-7.1 se recogen todos los grafos correspondientes a los Filtros FIR Paso bajo construidos con precisión 8 bit y 16 bit, en los que se ha corregido la presencia de nodos que sufrían compensación de errores.

Área Bloque Multiplicador y Área total de Filtro FIR Paso bajo BM DetectError.

Mediante el uso de la aplicación **DetectError** se ha procedido a determinar la superficie de circuito necesario para implementar todos los Filtros FIR Paso bajo sin compensación de errores provisto de nodo detector de Soft Errors, que operan en las mismas condiciones descritas en 4.2.

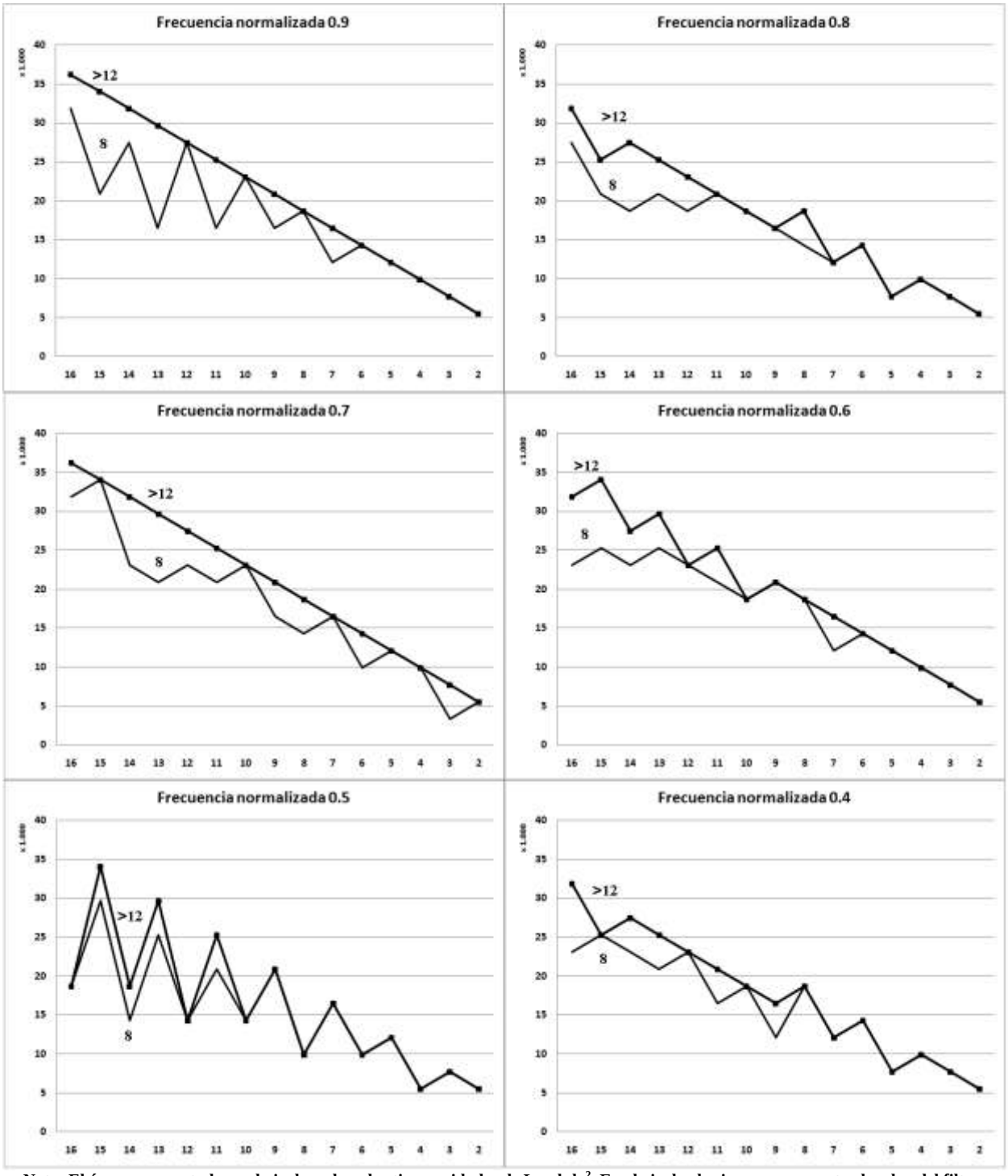
- Orden de filtro: 2 a 16.
- Frecuencia normalizada de corte: 0’1 a 0’9 en intervalos de 0’1.
- Precisión del Bloque Multiplicador: 8, 12, 16, 20 y 24 bit.

En las tablas 4.18 y 4.19 se indica la aportación del Registro Check y en las tablas 4.20 a 4.29 la contribución del Bloque Multiplicador y el área total del circuito lógico (suma de todos los componentes) para Filtros FIR Paso bajo de orden comprendido entre 2 y 16 y precisión de cálculo comprendida entre 8 y 24 bit.

En las figuras 4.7 y 4.8, se representan para cada frecuencia normalizada de corte del filtro, los valores de áreas del Registro Check para todos los Filtros FIR indicados.

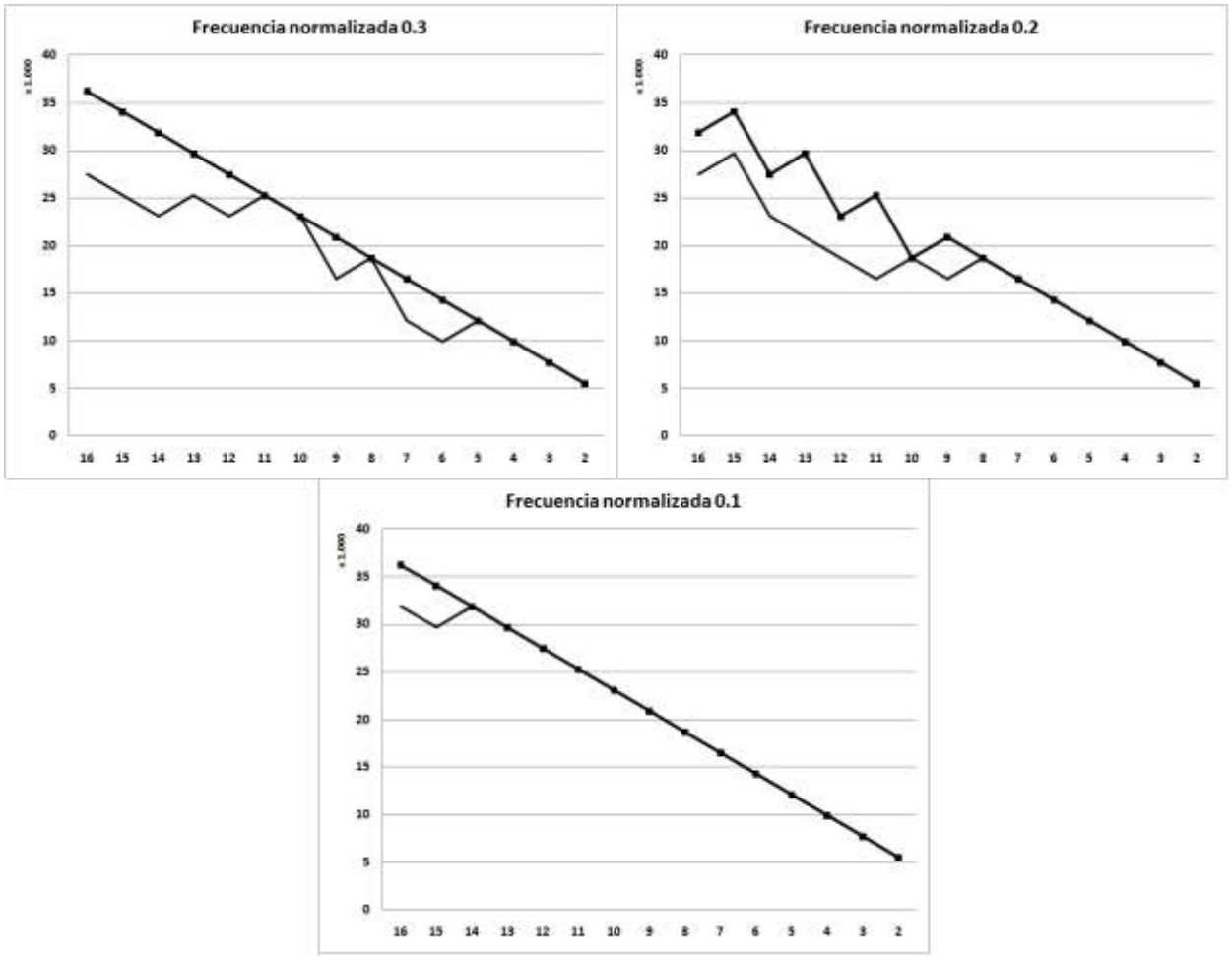
Como puede comprobarse se confirma el patrón teórico de necesidad de mayores áreas de circuito cuanto mayor es el orden del Filtro FIR y cómo, a igualdad de orden, es mayor la superficie cuanto más elevada es la precisión de cálculo, debido a la complejidad creciente de sus Registros Check.

En las figuras 4.9, 4.10 y 4.11 se representan los valores de área ocupadas por el Bloque Multiplicador y el área total de circuito para todos los Filtros FIR indicados.



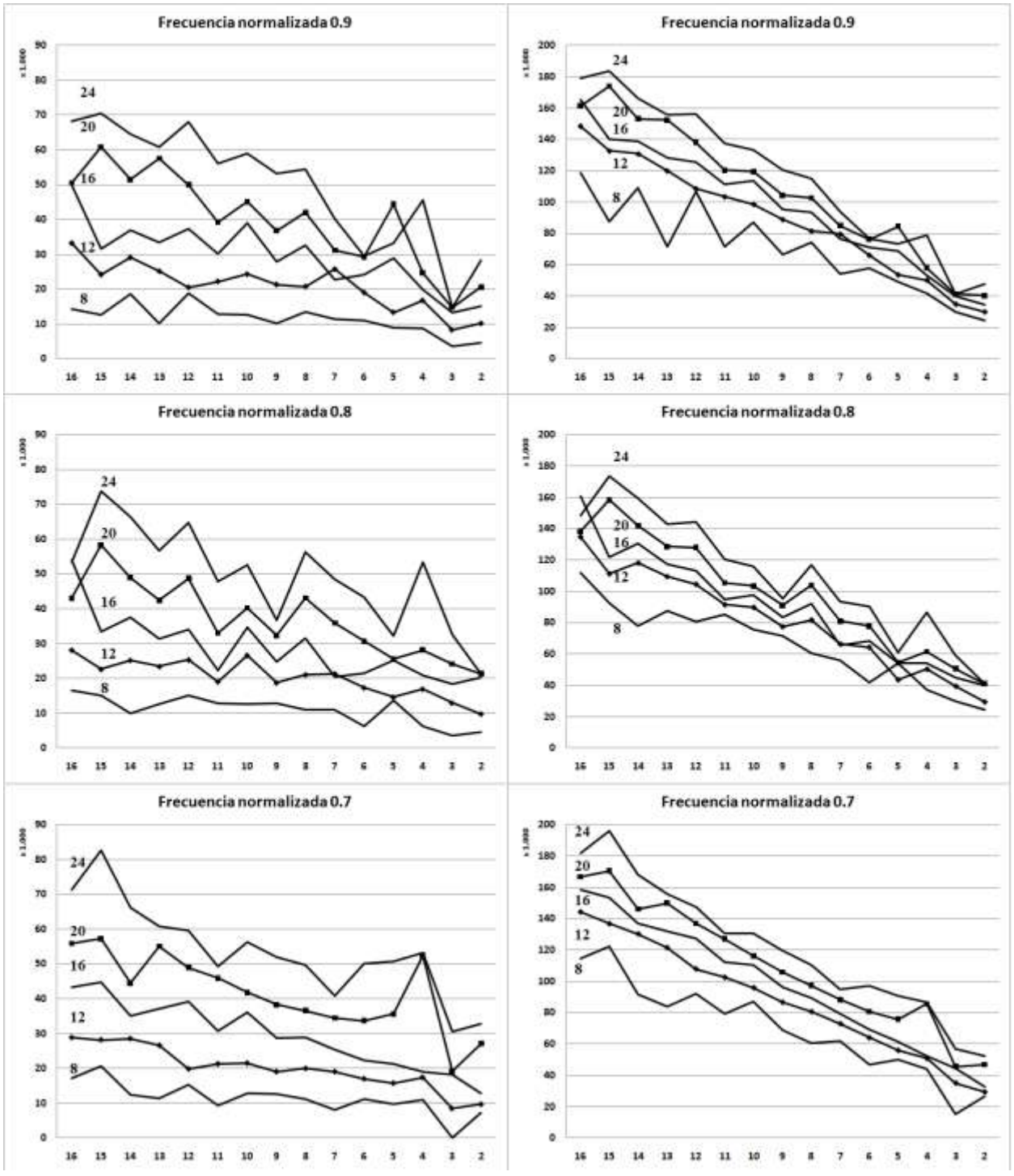
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, 8 y mayor de 12 bit, se indica en las propias gráficas

Figura 4.7 Área de Registro Check en Filtro FIR Paso bajo (DetectError)



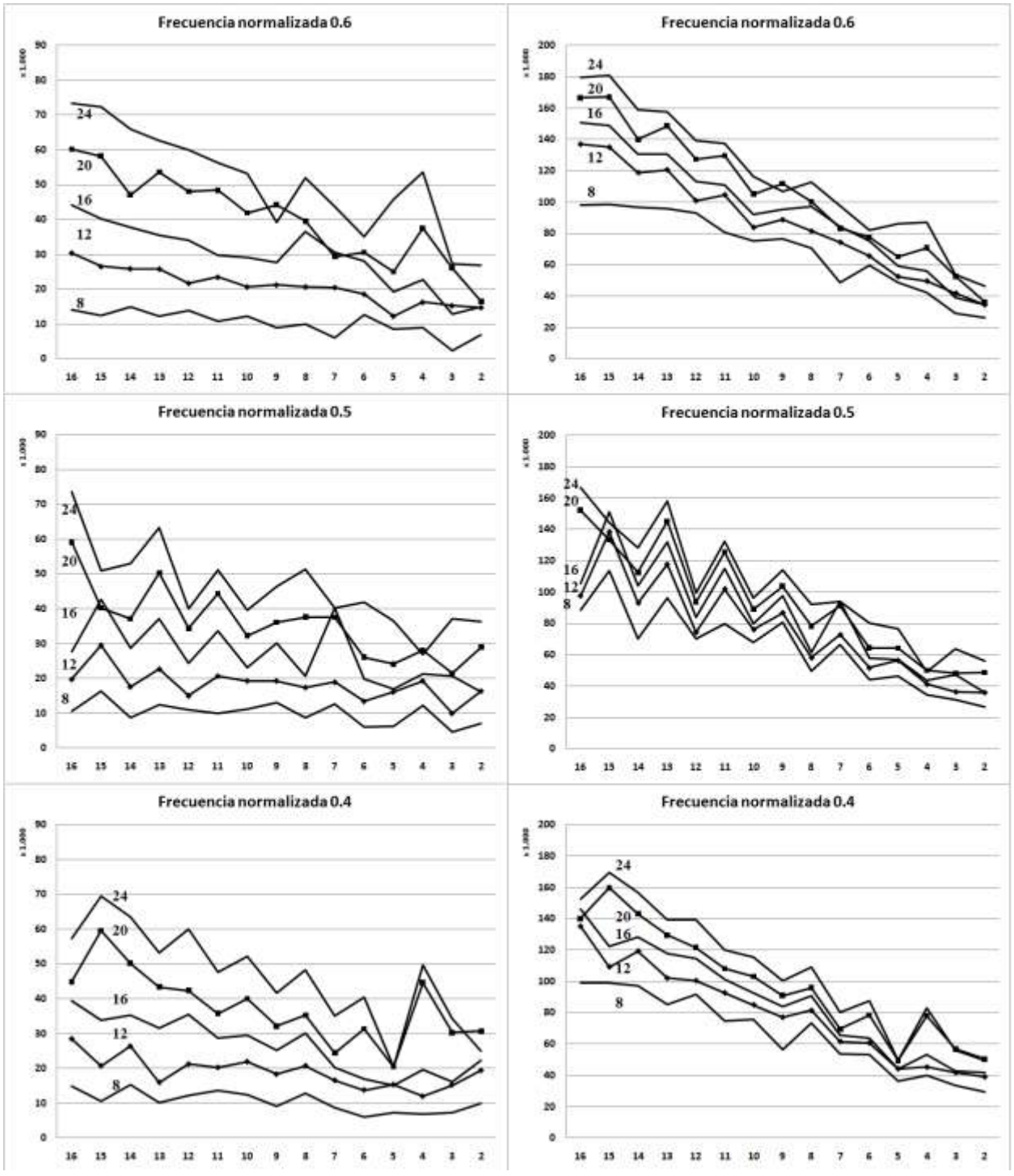
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, 8 y mayor de 12 bit, se indica en las propias gráficas

Figura 4.8 Área de Registro Check en Filtro FIR Paso bajo (DetectError)



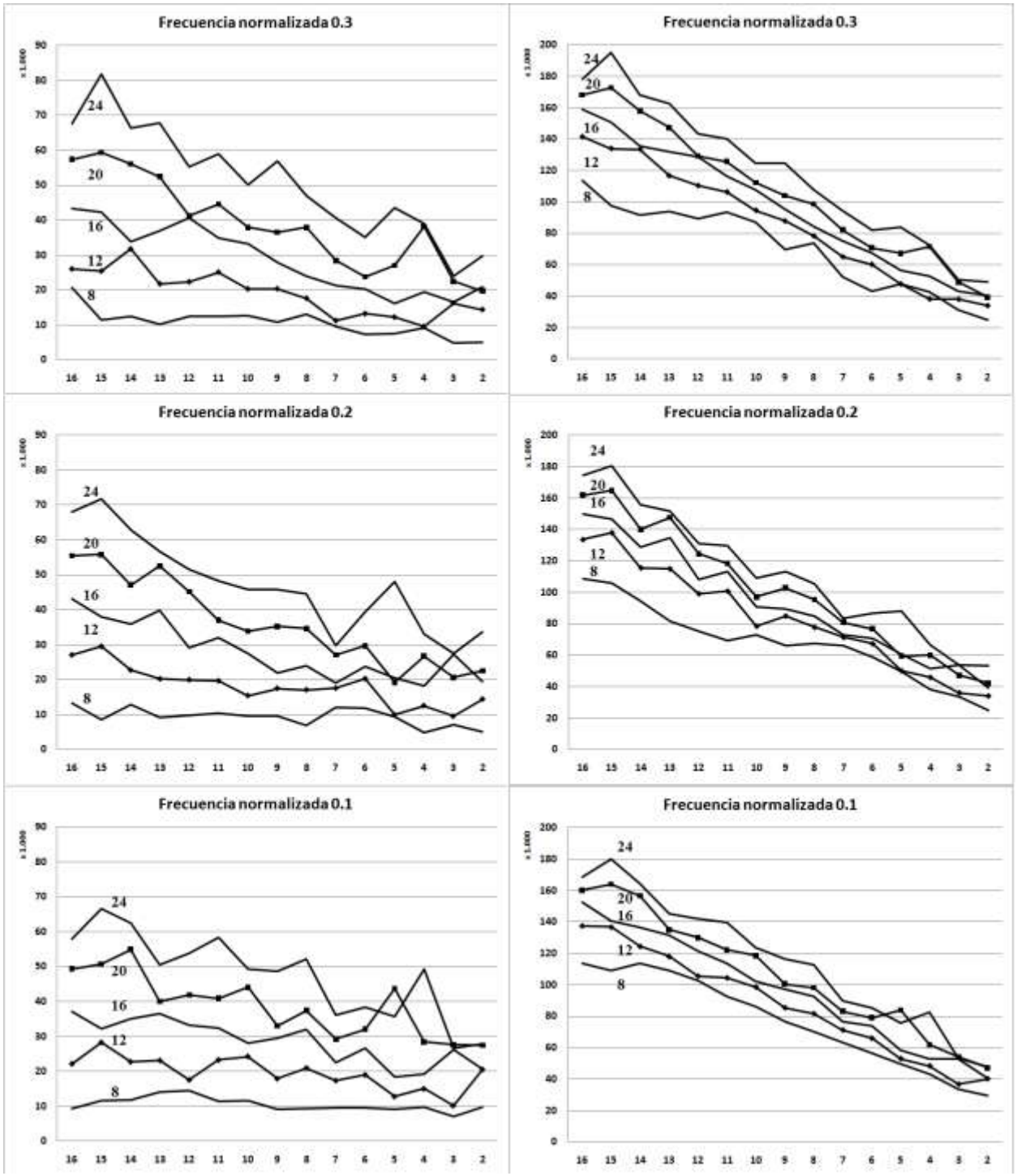
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.9 Área Bloque Multiplicador y Área total Filtro FIR Paso bajo (DetectError) (1)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.10 Área Bloque Multiplicador y Área total Filtro FIR Paso bajo (DetectError) (2)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.11 Área Bloque Multiplicador y Área total Filtro FIR Paso bajo (DetectError) (3)

El porcentaje de área aportado al total del circuito por el Registro Check y el Bloque Multiplicador varía desde una media de aproximadamente el 40% para cálculos con 8 bit de precisión a una media de más del 60% para precisión de 24 bit.

En la figura 4.12 se representa dicha variación según la precisión de cálculo utilizada.

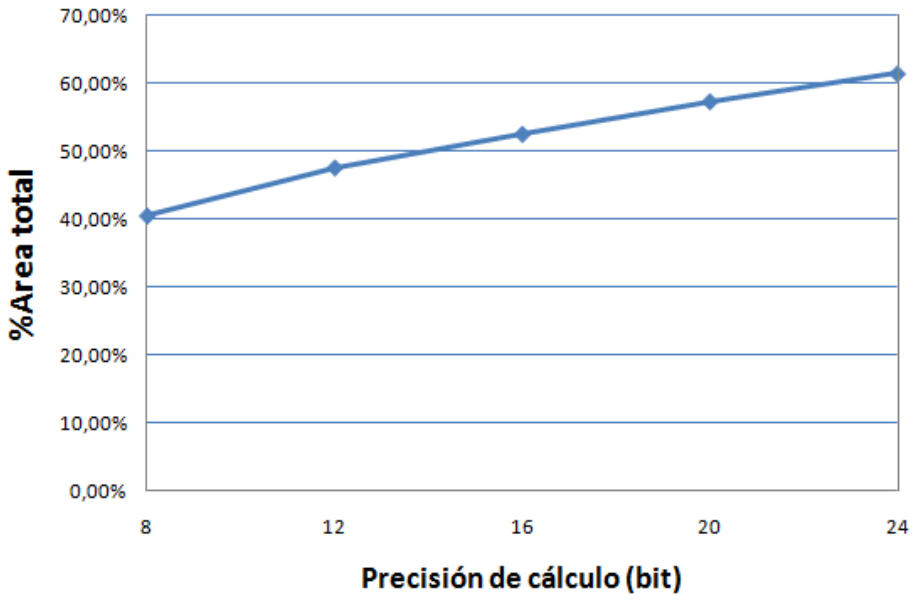


Figura 4.12 Porcentaje Área aportada por Bloque Multiplicador FIR Paso bajo (Spiral)

4.2.3 Área de Bloque Multiplicador DMR y Área total de circuito de Filtros FIR-DMR Paso bajo estándar (Bloque Spiral y tratamiento de errores).

La estrategia de detección de Soft Error de las implementaciones DMR (Redundancia Dual Modular) consiste en comparar las salidas proporcionadas por dos Bloques Multiplicadores que realizan la misma función simultáneamente.

Cada uno de los Bloques Multiplicadores estándar, recibe su entrada y procesa, de acuerdo a su orden, N+1 salidas, que entran en el Comparador, pero solo las salidas de uno de ellos, el llamado *Master*, ingresan en los respectivos Registros de Acumulación para generar la Salida del Filtro digital.

De acuerdo con ello, el esquema de un Filtro FIR DMR sería a grandes rasgos el indicado en la figura 4.13.

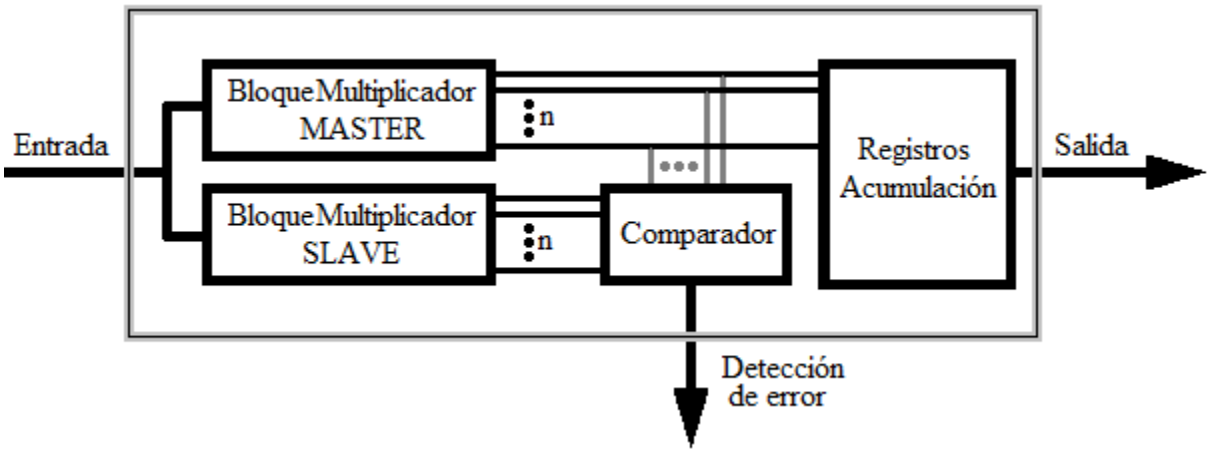


Figura 4.13 Esquema de un Filtro FIR con Redundancia Dual Modular.

Cuando el comparador detecta que los valores de salida proporcionados por ambos Bloques Multiplicadores son diferentes, confirma la existencia de un Soft Error.

Sea AREA_DMR el área del conjunto formado en un Filtro FIR-DMR por los dos Bloques Multiplicadores estándar más el Comparador.

Podemos calcular su valor aproximado mediante la siguiente expresión.

AREA_DMR = 2 * AREA_BM_Spiral + AREA_Comparador

El Área ocupada por el Comparador DMR puede estimarse de acuerdo con:

AREA_ Comparador = 0’5 * (AREA_RegistroCheck + 0’5 * AreaSumador)

Área DMR y Área total de Filtro FIR DMR Paso bajo BM BM Spiral.

En las Tablas 4.30 a 4.41 se presentan los datos obtenidos:

Área Comparador Filtro FIR DMR:	Tablas 4.30 y 4.31,
Área Bloque Multiplicador Filtro FIR DMR:	Tablas 4.32, 4.34, 4.36, 4.38, 4.40
Área total del circuito lógico Filtro FIR DMR:	Tablas 4.33, 4.35, 4.37, 4.39, 4.41

El cálculo de cada Área(Wn,N), correspondiente a cada frecuencia de corte y orden del Filtro, se determina a partir de las siguientes expresiones:

Filtros FIR DMR Paso bajo de precisión 8 bit

- Área Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) . Tabla 4.1
- Área Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) + 548’85.Tabla 4.18.

Filtros FIR DMR Paso bajo de precisión 12 bit

- Área Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) Tabla 4.3
- Área Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) + 548’85.Tabla 4.19.

Filtros FIR DMR Paso bajo de precisión 16 bit

- Área Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) Tabla 4.5
- Área Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) + 548’85.Tabla 4.19.

Filtros FIR DMR Paso bajo de precisión 20 bit

- Área Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) Tabla 4.7
- Área Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) + 548’85.Tabla 4.19.

Filtros FIR DMR Paso bajo de precisión 24 bit

- Área Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) Tabla 4.9
- Área Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) + 548’85.Tabla 4.19.

El porcentaje de área aportado al total del circuito FIR DMR Paso bajo por el comparador, disminuye a medida que utilizamos filtros con orden más elevado, desde una media de aproximadamente el 13% para cálculos con 8bit de precisión a una media en torno al 7% para precisión de 24 bit.

El tamaño del comparador no depende de la precisión utilizada sino del número de salidas del Bloque Multiplicador, pero como el resto de componentes si se ven afectados por dicha variable, a medida que aumenta la precisión de cálculo la aportación del comparador es cada vez menor.

En la figura 4.14 se representa la aportación del comparador al área total del circuito según la precisión de cálculo utilizada.

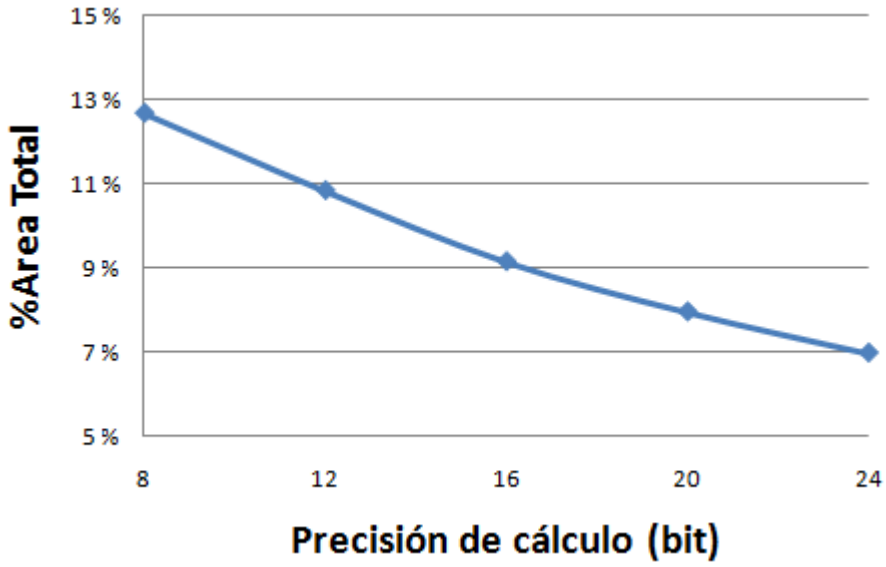
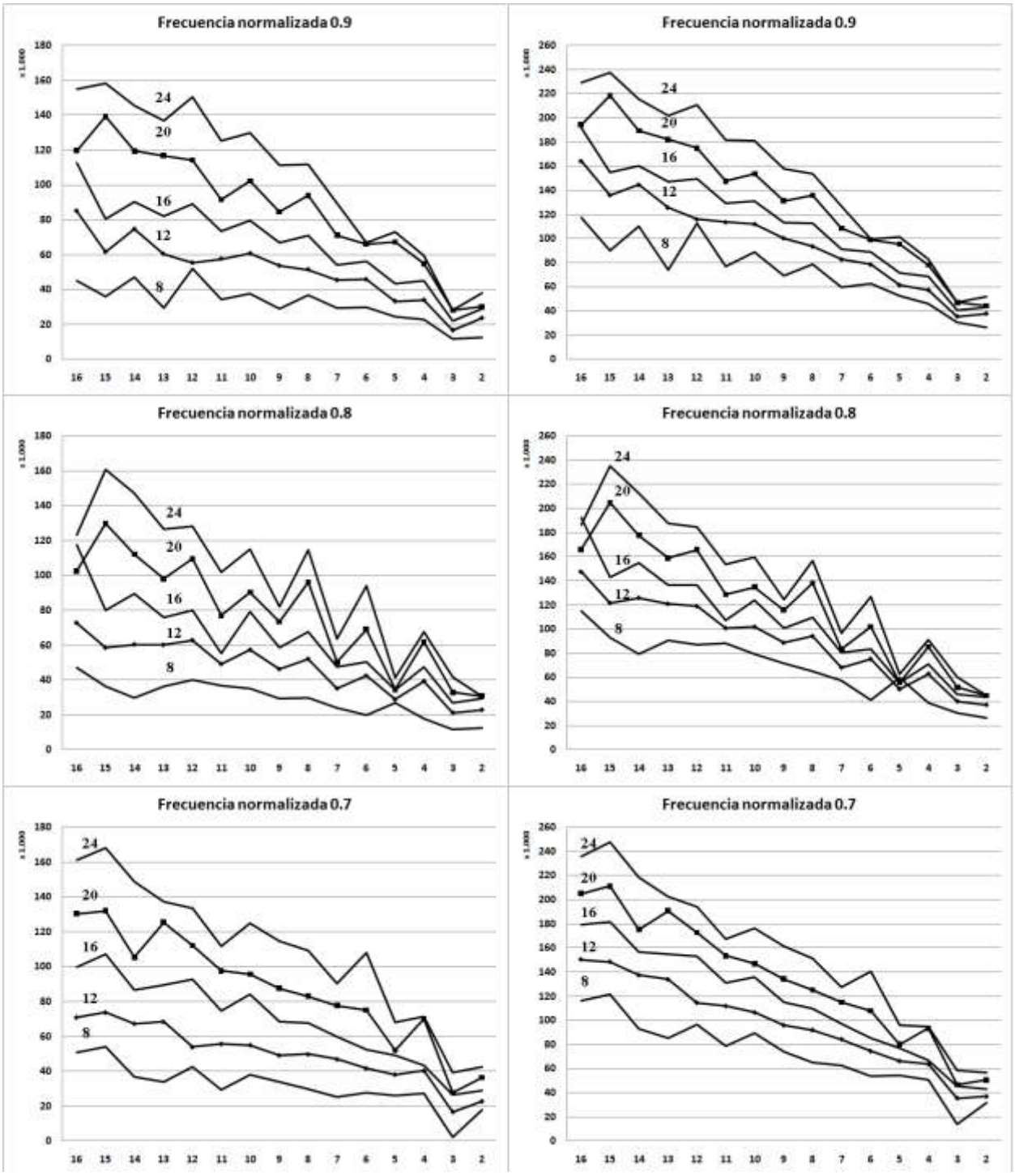


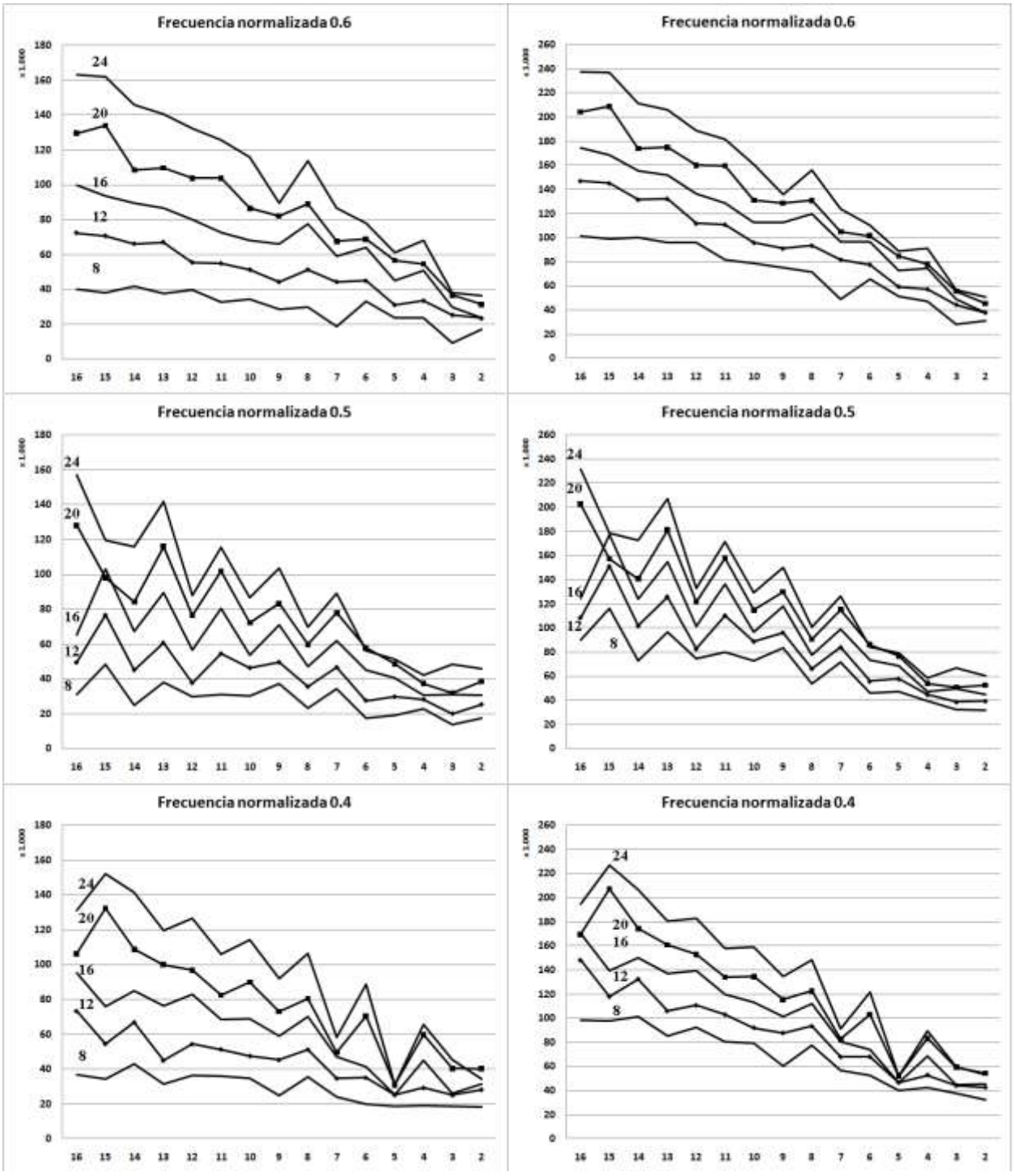
Figura 4.14 Porcentaje del Área aportada por el Comparador FIR DMR Paso bajo (Spiral)

En las figuras 4.15, 4.16 y 4.17, se representan para cada frecuencia normalizada de corte y para Filtros FIR-DMR Paso bajo con precisión de cálculo comprendida entre 8 y 24 bit y orden de Filtro comprendido entre 2 y 16, los valores de áreas del Bloque Multiplicador DMR, conjunto formado por los dos Bloques Multiplicadores estándar más su Comparador, y los valores de área total para implantar todo el circuito lógico.



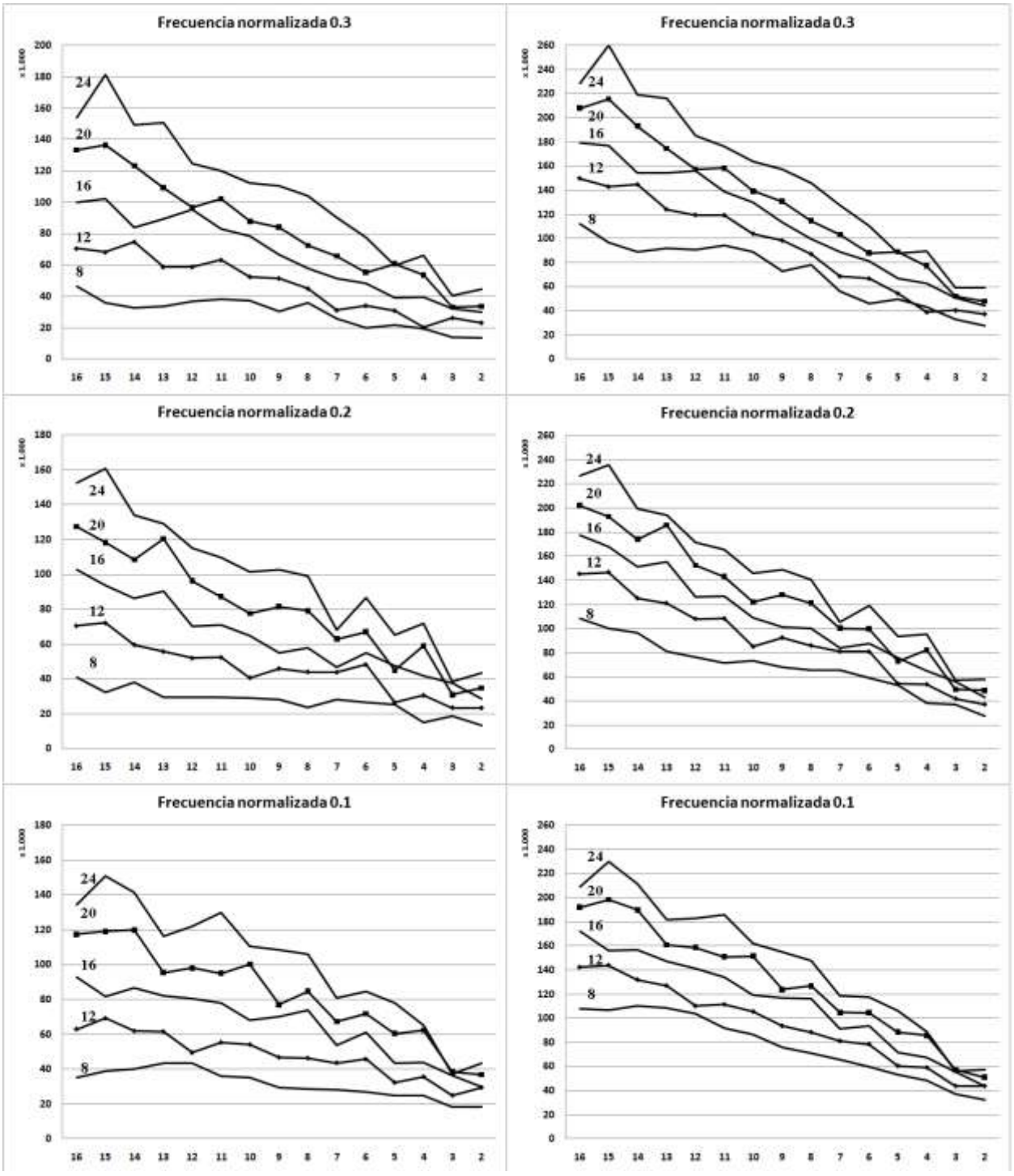
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.15 Área Bloque Multiplicador/Comparador y Área total Filtro DMR Paso bajo (1).



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.16 Área Bloque Multiplicador/Comparador y Área total Filtro DMR Paso bajo (2).



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro La precisión de cálculo, se indica en las propias gráficas.

Figura 4.17 Área Bloque Multiplicador/Comparador y Área total Filtro DMR Paso bajo (3).

El porcentaje de área aportado al total del circuito por los dos Bloques Multiplicadores y el Comparador DMR, varía desde una media de aproximadamente el 45% para cálculos con 8 bit de precisión a una media del 70% para precisión de 24 bit.

En la figura 4.18 se representa dicha variación según la precisión de cálculo utilizada.

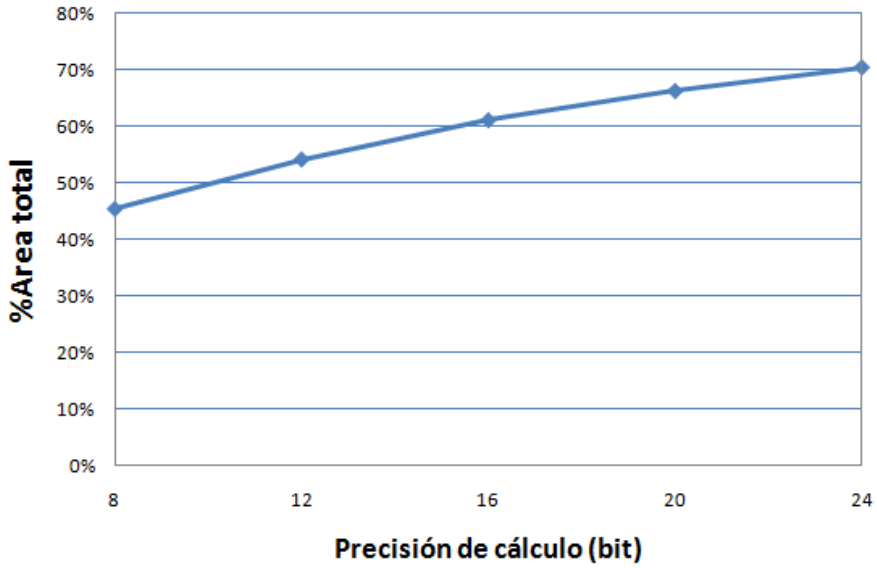


Figura 4.18 Porcentaje Área aportada por Bloque Multiplicador FIR DMR Paso bajo (Spiral)

4.3 Cálculo de superficie de Filtros FIR Paso alto con detección de Soft Errors.

En este apartado se procederá al estudio del comportamiento de Filtros FIR Paso alto, determinando la superficie necesaria para implementar en un circuito ASIC, el Bloque Multiplicador, los Registros de Acumulación y el Registro Check de un Filtro FIR con Bloque DetectError, que operen en las siguientes condiciones:

- Orden de filtro: Orden par. Valores entre 2 y 16.
- Frecuencia normalizada de corte: 0’1 a 0’9 en intervalos de 0’1.
- Precisión del Bloque Multiplicador: 8, 12, 16, 20 y 24 bit.

Además, para confirmar que DetectError presenta ventajas sobre otras estrategias con y sin tratamiento de errores, se determinará también dicha superficie para Filtros FIR que operen con Bloque Multiplicador estándar (Bloque Spiral) y Filtros FIR con Redundancia Dual Modular (Bloque Spiral y tratamiento de errores).

Para conseguir dicha información se han considerado las siguientes etapas:

4.3.1) Obtención de grafos, área de Bloque Multiplicador y área total de circuito de Filtros FIR Paso alto estándar (Bloque Spiral).

4.3.2) Obtención de grafos, área de Bloque Multiplicador que incluye nodo detector de errores y área total de circuito de Filtros FIR Paso alto con nodos replicados (Bloque DetectError).

4.3.3) Obtención del área del Bloque Multiplicador que incluye comparador para detección de errores y área total de circuito de Filtros FIR-DMR Paso alto (Bloque Spiral y tratamiento de errores).

4.3.1 Obtención de grafos, Área de Bloque Multiplicador y Área total de circuito de Filtros FIR Paso alto estándar (Bloque Spiral).

En el Anexo A-6.3 se recogen, a modo de ejemplo, tan solo los grafos correspondientes a dos situaciones concretas:

*) Filtro FIR Paso alto. Orden 4, Frecuencia normalizada de corte 0.8 y precisión 8 bit

*) Filtro FIR Paso alto. Orden 6, Frecuencia normalizada de corte 0.5 y precisión 16 bit

Área Bloque Multiplicador y Área total de Filtro FIR Paso alto (Bloque Spiral).

En las tablas 4.42 a 4.51 se presentan los datos obtenidos mediante Spiral, de la superficie de circuito necesario para implementar los Filtro FIR Paso alto descritos en el punto 4.3, indicando por separado la contribución del Bloque Multiplicador y la del área total del circuito lógico.

Como puede comprobarse se confirma el patrón teórico de necesidad de mayores áreas de circuito cuanto mayor es el orden del Filtro FIR Paso alto y como a igualdad de orden es mayor la superficie cuanto más elevada es la precisión de cálculo, debido a la complejidad creciente de sus Bloques Multiplicadores.

El porcentaje de área aportado al total del circuito por los Bloques Multiplicadores, varía desde una media de aproximadamente el 26.5% para cálculos con 8bit de precisión a una media de más del 54% para precisión de 24 bit.

En la figura 4.19 se representa dicha variación según la precisión de cálculo utilizada.

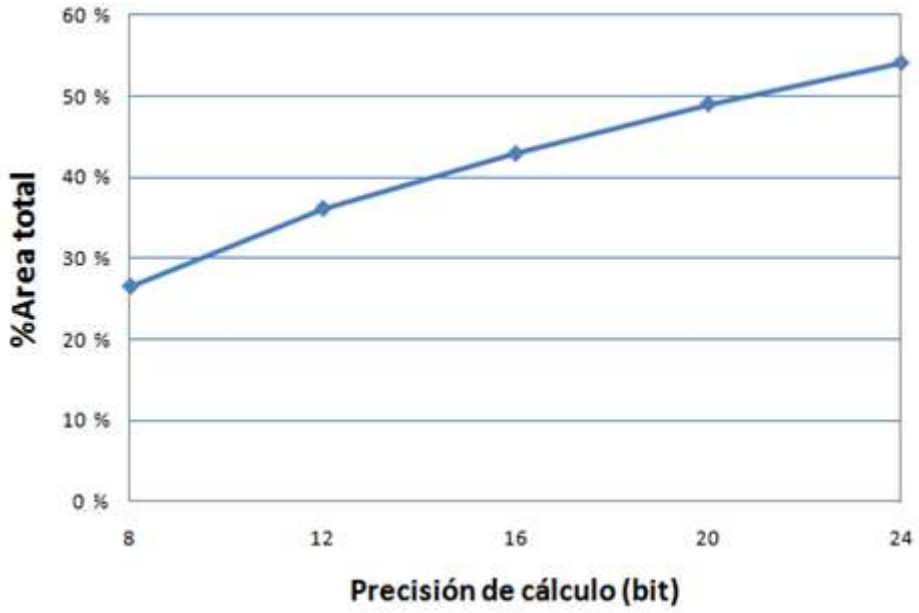
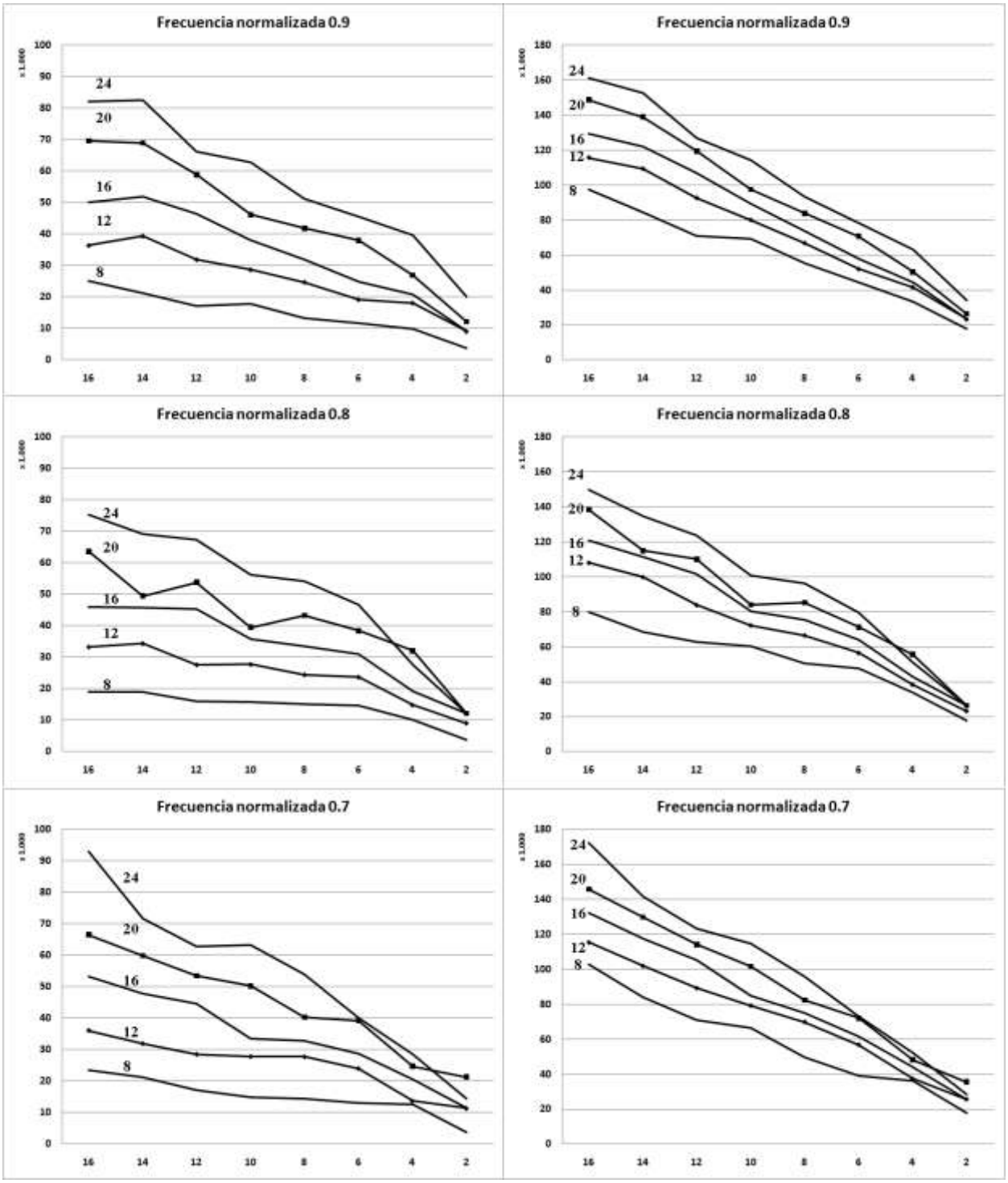


Figura 4.19 Porcentaje del Área aportada por Bloque Multiplicador FIR Paso alto (Spiral)

En las tablas 4.52 y 4.53 se recogen los datos de Coeficientes nulos en los Bloques Multiplicadores de Filtros FIR Paso alto calculados con precisión 8 bit y precisión mayor que 12 bit. Se repite el hecho de que para frecuencia de corte normalizada 0´5 hay un gran número de coeficientes nulos.

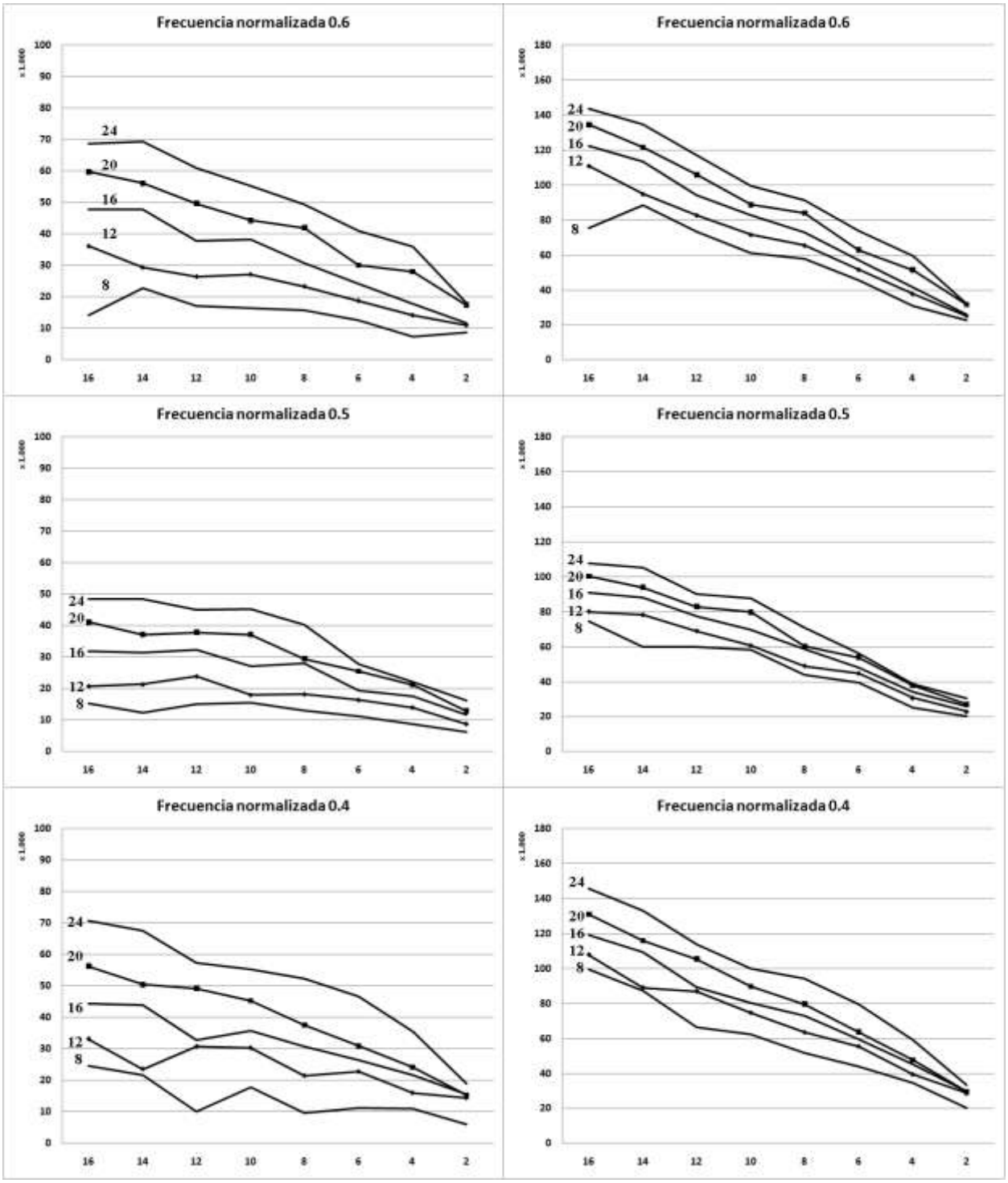
No se observa un comportamiento errático tan claro como el descrito en el caso de los Filtros Paso bajo, cuyos filtros de orden par tenían más coeficientes nulos que los impares ya que los Filtros Paso alto solo pueden construirse de orden par, pero si aparecen irregularidades y en bastantes situaciones, sobre todo entre Filtros de orden 6-8 y 10-12, el área total de Filtros de orden mayor se mantiene en valores muy similares al de Filtros de orden más bajo.

En las figuras 4.20, 4.21 y 4.22 se representan para cada frecuencia normalizada de corte del filtro, los valores de áreas del Bloque multiplicador y Área total de circuito para Filtros FIR Paso alto con precisión de cálculo comprendida entre 8 y 24 bit.



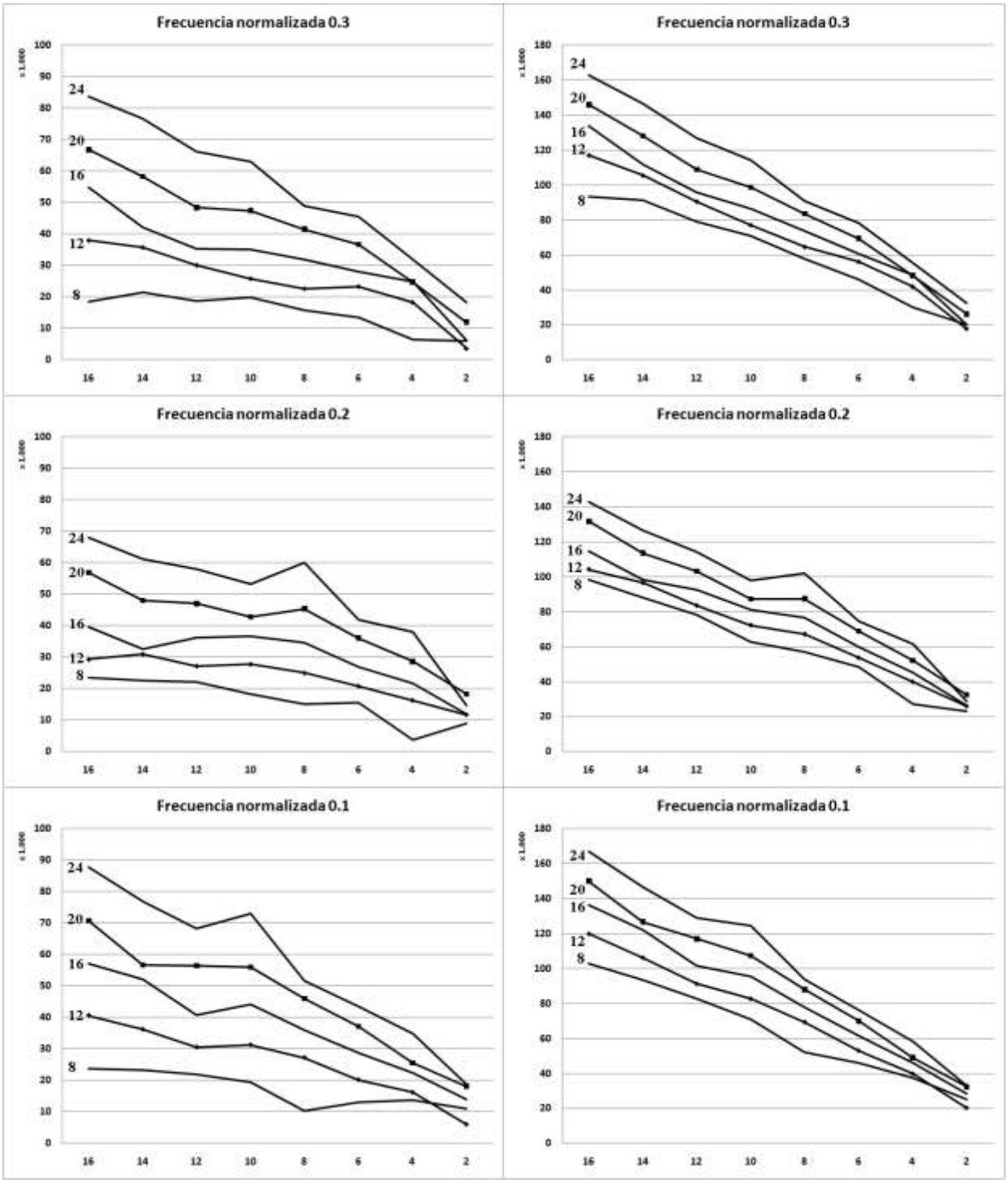
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.20 Área Bloque Multiplicador y Área total Filtro FIR Paso alto Spiral (1)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.21 Área Bloque Multiplicador y Área total Filtro FIR Paso alto Spiral (2)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.22 Área Bloque Multiplicador y Área total Filtro FIR Paso alto Spiral (3)

4.3.2 Obtención de grafos, Área de Bloque Multiplicador y Área total de circuito de Filtros FIR Paso alto (Bloque DetectError).

Tras analizar los datos Verilog de todos los Filtros FIR Paso alto que operan en las condiciones indicados en el punto 4.3, se comprueba que solo es necesario realizar Replicaciones Parciales en los casos indicados con la letra "R" en las tablas 4.54 y 4.58. En todas ellas, se expresa con el cardinal adecuado el número de nodos que es necesario replicar para conseguir eliminar la compensación de errores.

En el Anexo A-7.2 se recogen todos los grafos correspondientes a los Filtros FIR Paso alto construidos con precisión 8 bit y 16 bit, en los que se ha detectado la presencia de compensación de errores.

Área Bloque Multiplicador y Área total de Filtro FIR Paso alto BM DetectError.

Mediante el uso de **DetectError** se ha procedido a determinar la superficie de circuito necesario para implementar todos los Filtros FIR Paso alto sin compensación de errores provisto de nodo detector de Soft Errors, que operan con las mismas condiciones descritas en 4.3.

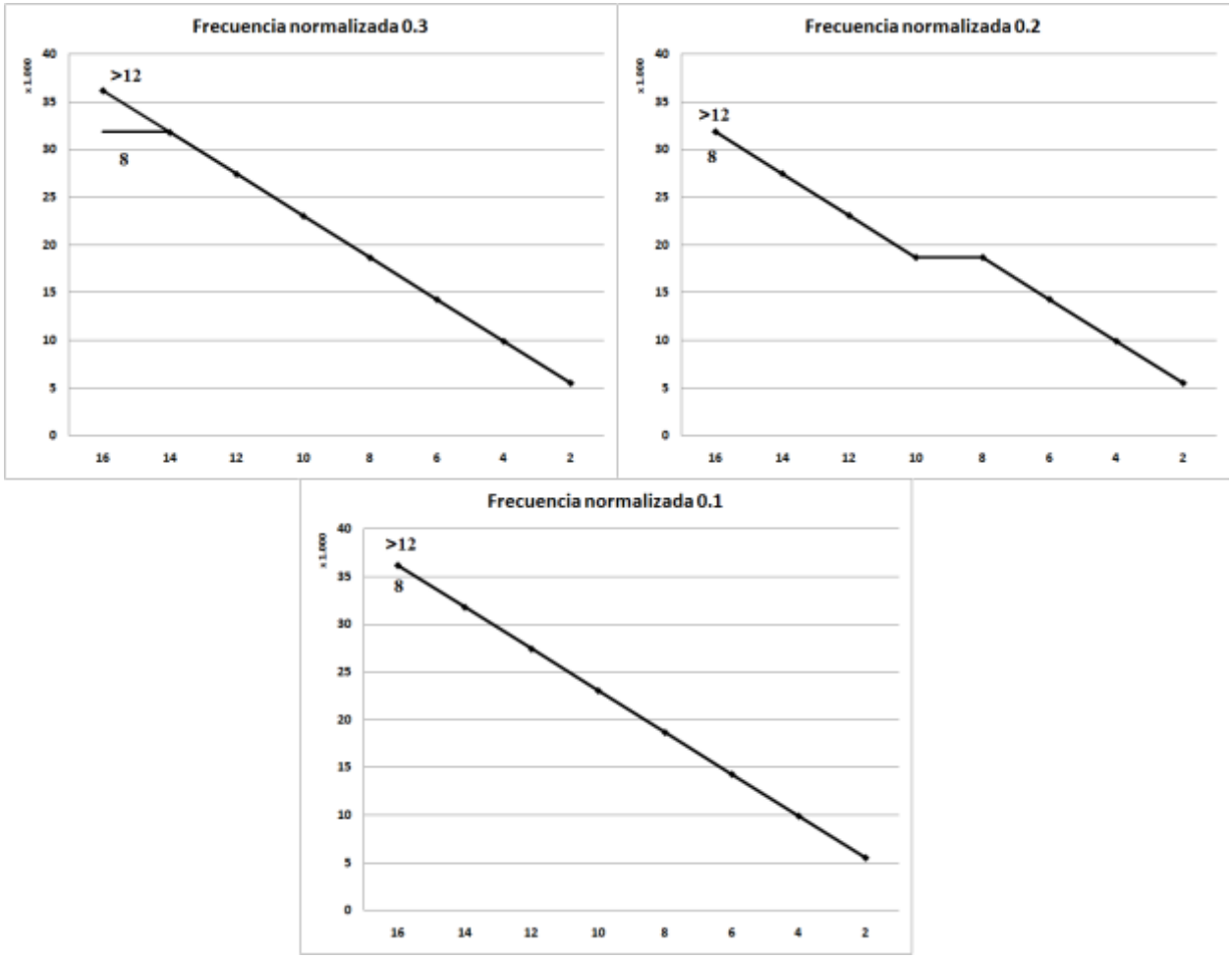
- Orden de filtro: Orden par. Valores comprendidos entre 2 y 16.
- Frecuencia normalizada de corte: 0'1 a 0'9 en intervalos de 0'1.
- Precisión del Bloque Multiplicador: 8, 12, 16, 20 y 24 bit.

En las tablas 4.59 y 4.60 se indica la aportación del Registro Check y en las tablas 4.61 a 4.70, la contribución del Bloque Multiplicador y el área total del circuito lógico (suma de todos los componentes) de dichos Filtros.

En las figuras 4.23 y 4.24, se representan para cada frecuencia normalizada de corte del filtro, los valores de áreas del Registro Check para todos los Filtros FIR Paso alto de 8 bit y mayor de 12 bit de precisión de cálculo y orden comprendido entre 2 y 16.

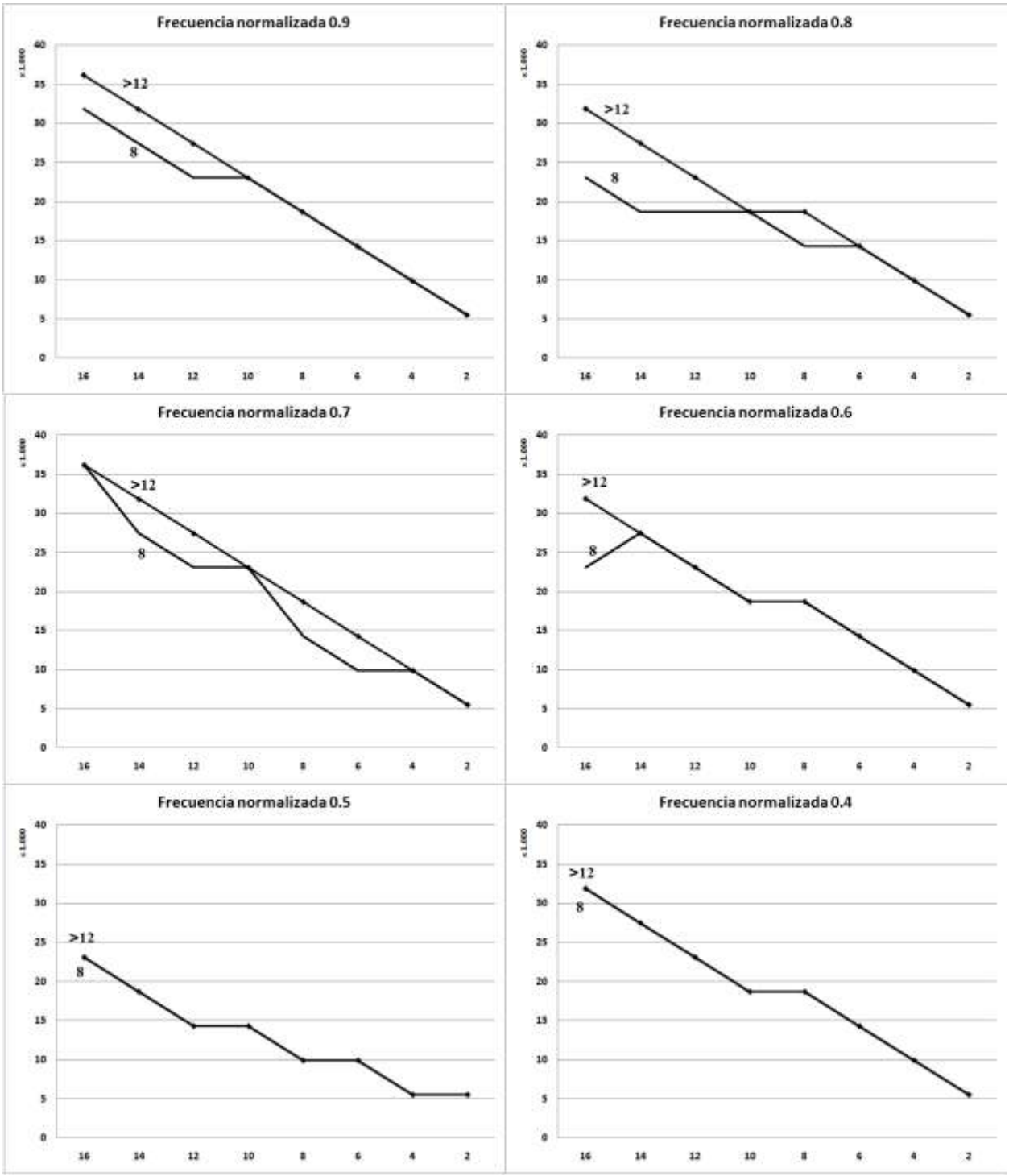
En las figuras 4.25, 4.26 y 4.27 se representan los valores de área ocupada por el Bloque Multiplicador y el área total de circuito para todos los Filtros FIR Paso alto indicados en el punto 4.3

Como puede comprobarse, en todos los casos, se confirma de nuevo el patrón teórico de necesidad de mayores áreas de circuito cuanto mayor es el orden del Filtro FIR Paso alto y como a igualdad de orden es mayor la superficie cuanto más elevada es la precisión de cálculo, debido a la complejidad creciente de sus Bloques Multiplicadores y de sus Registros Check.



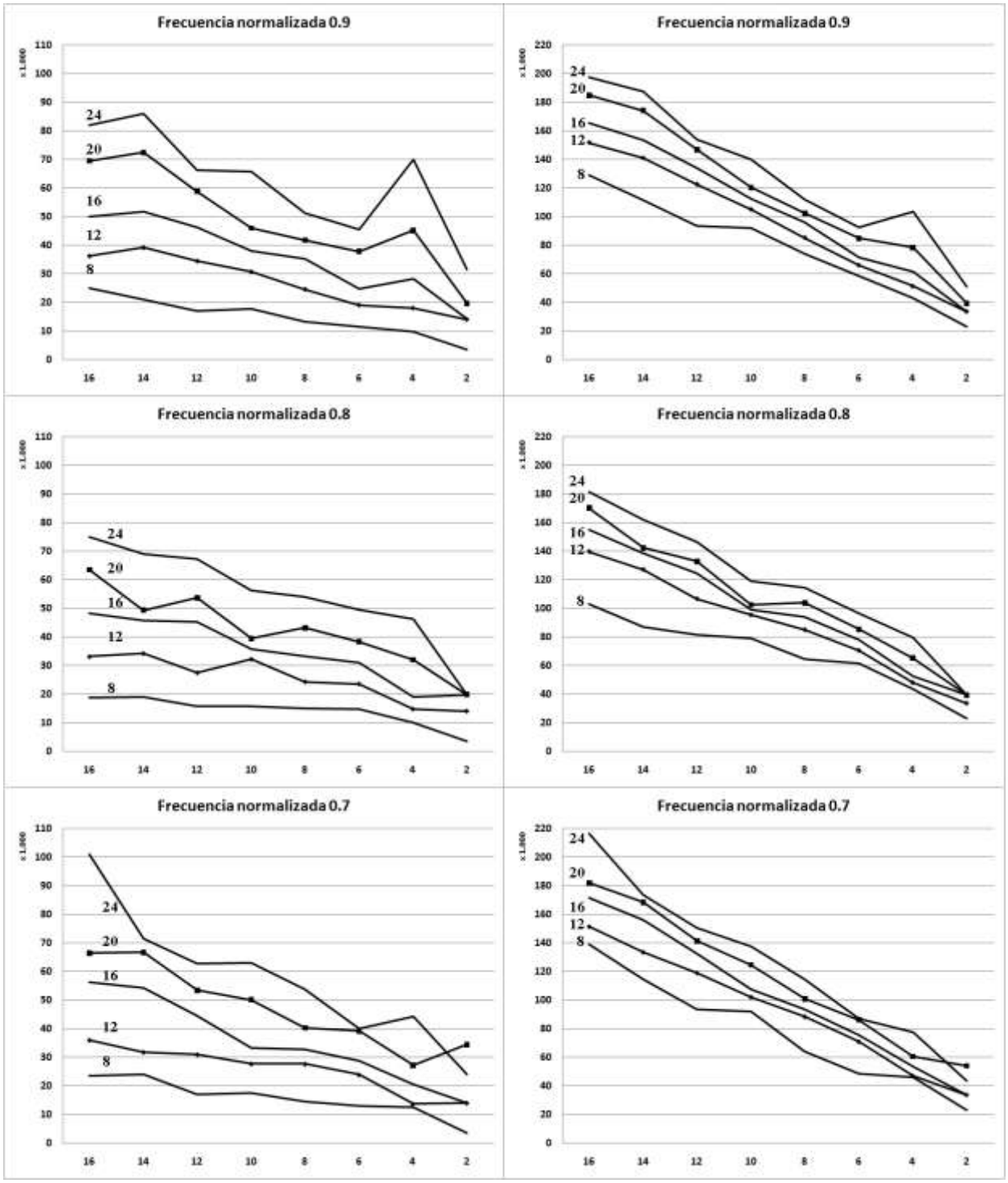
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, 8 y mayor de 12 bit, se indica en las propias gráficas

Figura 4.23 Registro Check 8 bit y mayor 12 bit



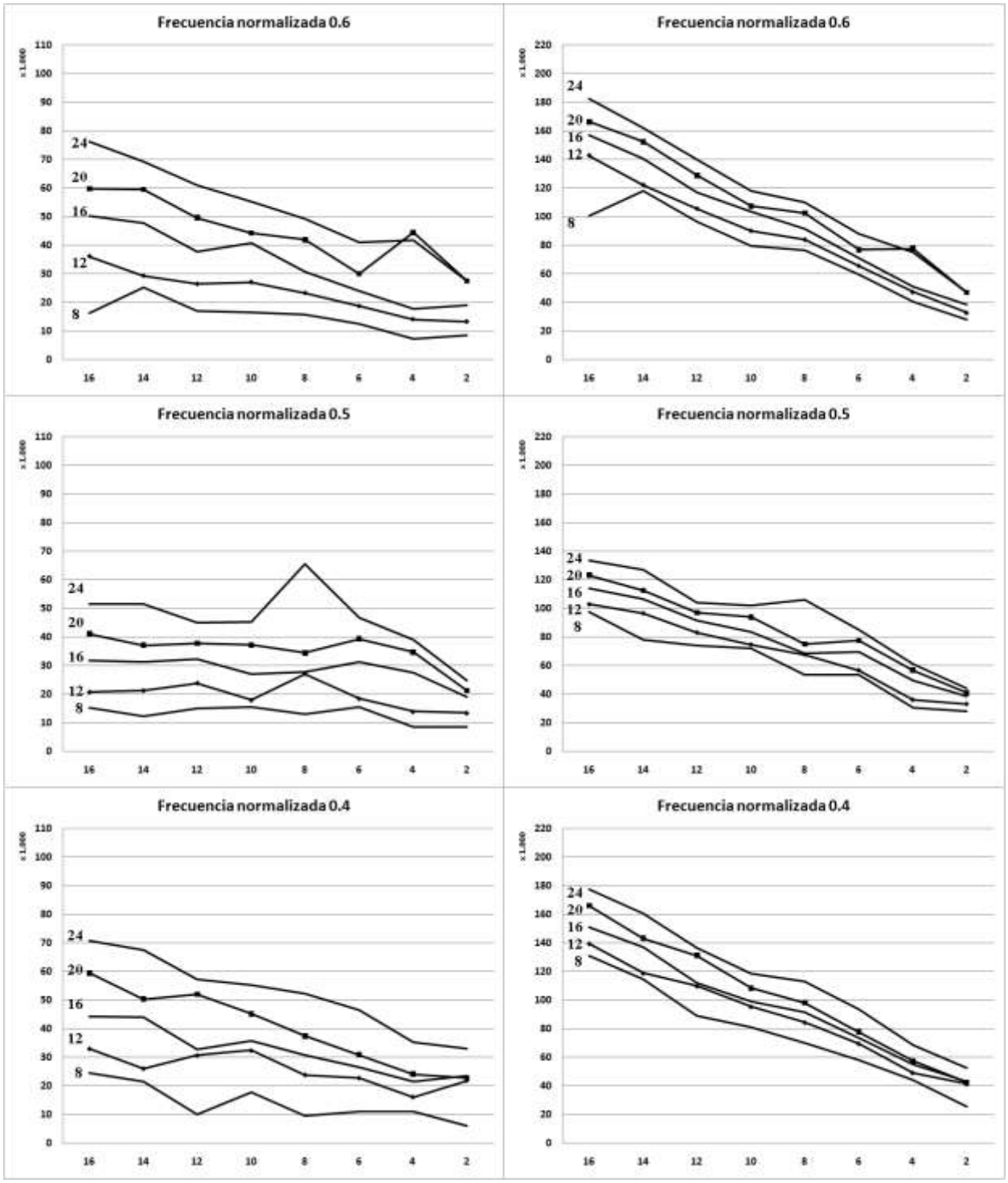
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, 8 y mayor de 12 bit, se indica en las propias gráficas

Figura 4.24 Registro Check 8 bit y mayor de 12 bit



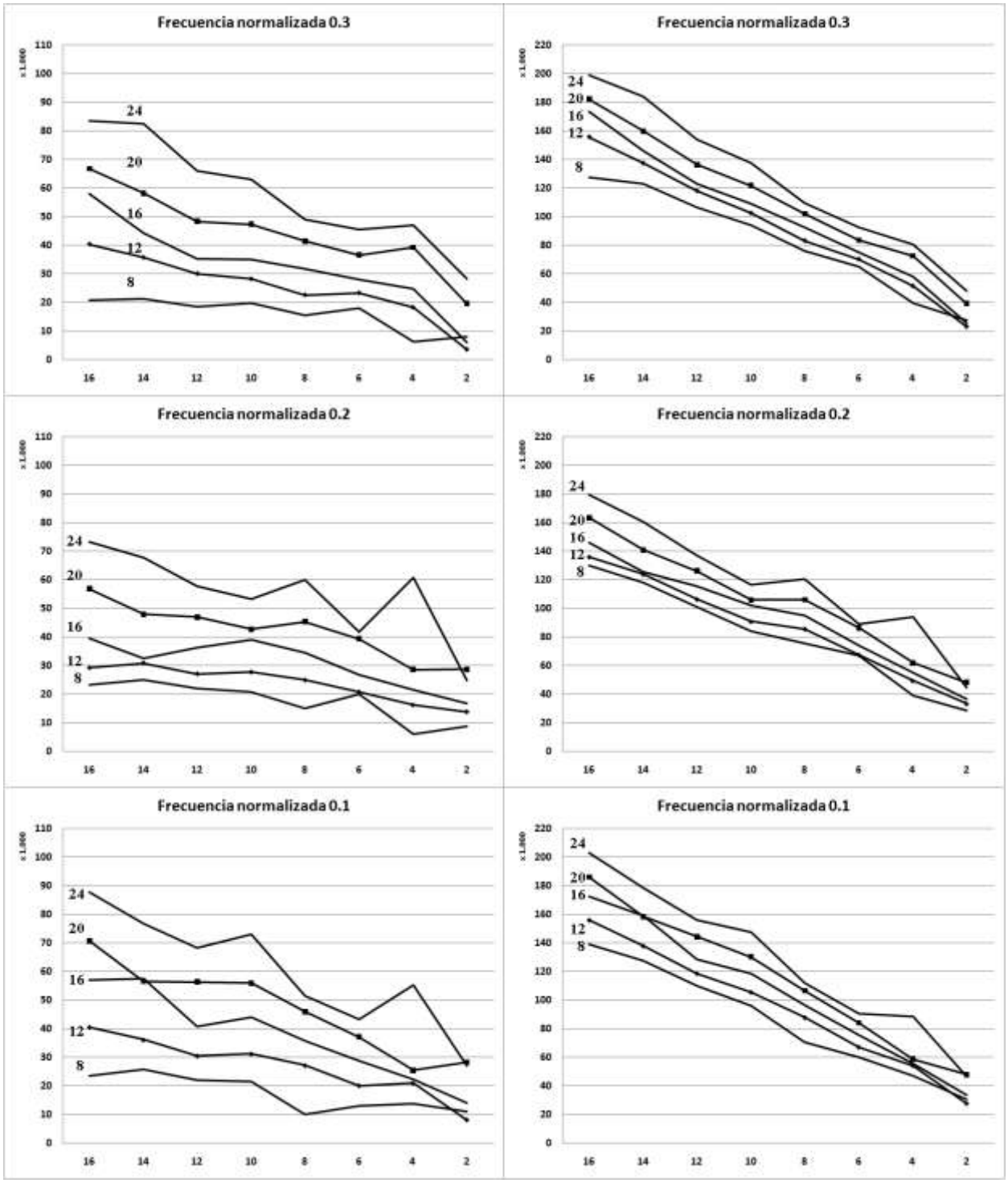
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo se indica en las propias gráficas

Figura 4.25 Área Bloque Multiplicador y Área total Filtro FIR Paso alto DetectError (1)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo se indica en las propias gráficas

Figura 4.26 Área Bloque Multiplicador y Área total Filtro FIR Paso alto DetectError (2)



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo se indica en las propias gráficas

Figura 4.27 Área Bloque Multiplicador y Área total Filtro FIR Paso alto DetectError (3)

El porcentaje de área aportado al total del circuito por el Bloque Multiplicador varía desde una media de aproximadamente el 45% para cálculos con 8 bit de precisión a una media de menos del 60% para precisión de 24 bit.

En la figura 4.28 se representa dicha variación según la precisión de cálculo utilizada.

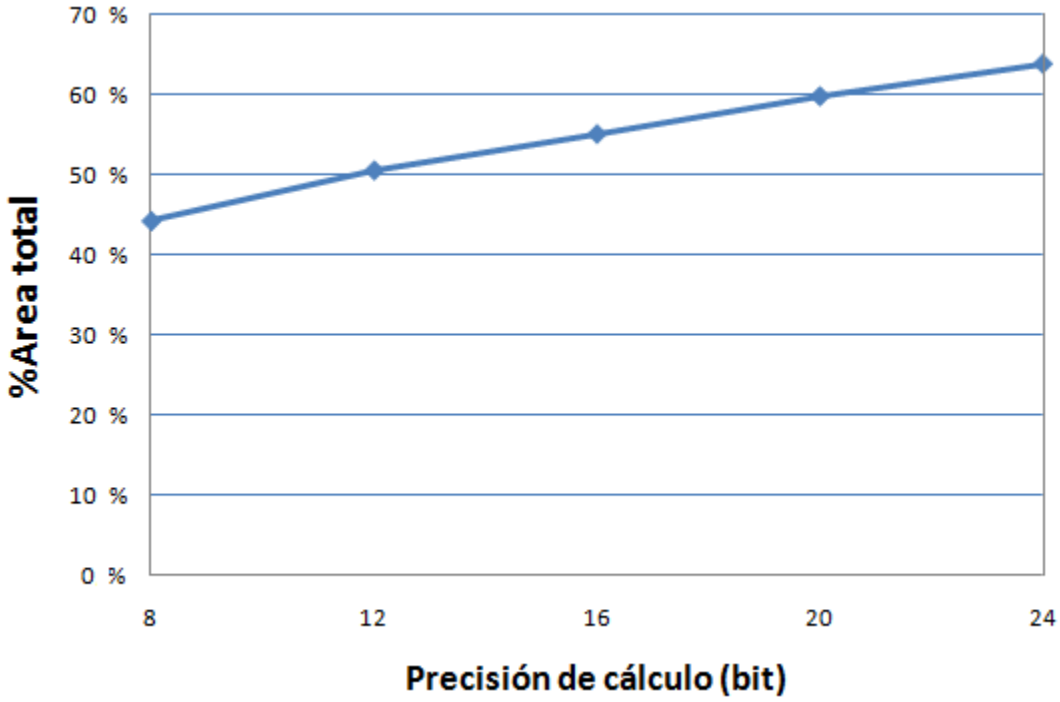


Figura 4.28 Porcentaje Área aportada por Bloque Multiplicador FIR Paso alto (DetectError)

4.3.3 Área de Bloque Multiplicador DMR y Área total de circuito de Filtros FIR-DMR Paso alto estándar (Bloque Spiral y tratamiento de errores).

Área DMR y área total de circuito FIR DMR Paso alto BM Spiral.

En las Tablas 4.71 a 4.82 se presentan los datos obtenidos:

Área Comparador Filtro FIR DMR Paso alto: Tablas 4.71 y 4.72,

Área Bloque Multiplicador Filtro FIR DMR: Tablas 4.73, 4.75, 4.77, 4.79, 4.81

Area total del circuito lógico Filtro FIRDMR: Tablas 4.74, 4.76, 4.78, 4.80, 4.82

El cálculo de cada Área (Wn,N), correspondiente a cada frecuencia de corte y orden del Filtro FIR DMR Paso alto, se determina a partir de las siguientes expresiones:

Filtros FIR DMR Paso alto de precisión 8 bit

- Area Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) de Tabla 4.42
- Area Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) de Tabla 4.59

Filtros FIR DMR Paso alto de precisión 12 bit

- Area Multiplicador DMR(Wn,N) = 2*AREA_BM_Spiral(Wn,N) de Tabla 4.43
- Area Comparador DMR(Wn,N) = 0’5*AREA_RegistroCheck(Wn,N) de Tabla 4.60

Filtros FIR DMR Paso alto de precisión 16 bit

- Area Multiplicador DMR(Wn,N) = 2 * AREA_BM_Spiral(Wn,N) de Tabla 4.44
- Area Comparador DMR(Wn,N) = 0’5 * AREA_RegistroCheck(Wn,N) de Tabla 4.60

Filtros FIR DMR Paso alto de precisión 20 bit

- Area Multiplicador DMR(Wn,N) = 2 * AREA_BM_Spiral(Wn,N) de Tabla 4.45
- Area Comparador DMR(Wn,N) = 0’5 * AREA_RegistroCheck(Wn,N) de Tabla 4.60

Filtros FIR DMR Paso alto de precisión 24 bit

- Area Multiplicador DMR(Wn,N) = 2 * AREA_BM_Spiral(Wn,N) de Tabla 5.46
- Area Comparador DMR(Wn,N) = 0’5 * AREA_RegistroCheck(Wn,N) de Tabla 5.60

El porcentaje de área aportado al total del circuito FIR DMR Paso alto por el comparador, disminuye a medida que utilizamos filtros con orden más elevado, desde una media de aproximadamente el 11,5% para cálculos con 8bit de precisión a una media en torno al 6% para precisión de 24 bit.

El tamaño del comparador no depende de la precisión utilizada sino del número de salidas del Bloque Multiplicador, pero como el resto de componentes si se ven afectados por dicha variable, a medida que aumenta la precisión de cálculo la aportación del comparador es cada vez menor.

En la figura 4.29 se representa la aportación del comparador al área total del circuito FIR DMR Paso alto según la precisión de cálculo utilizada.

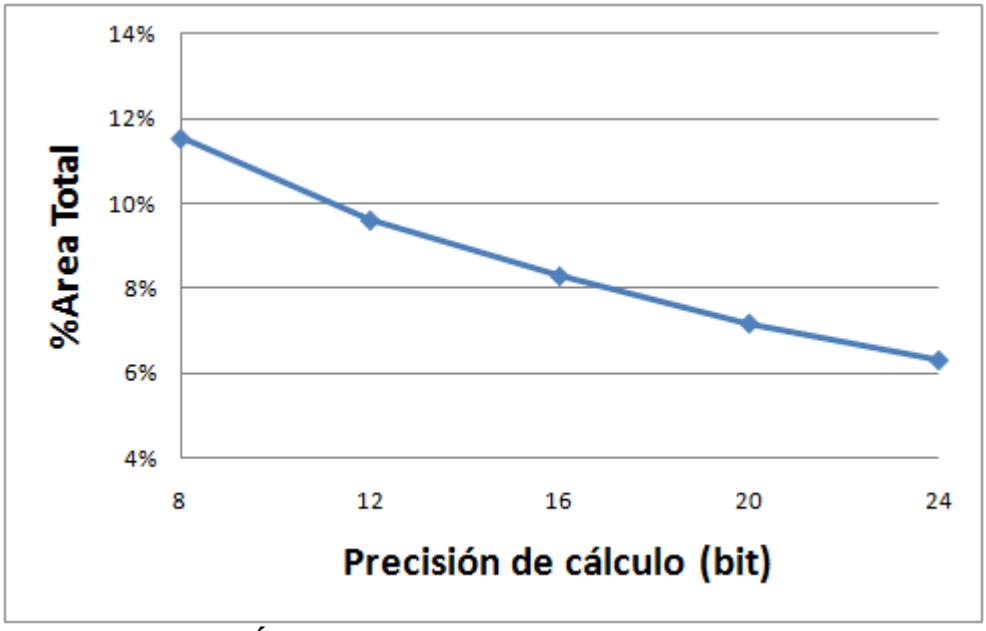
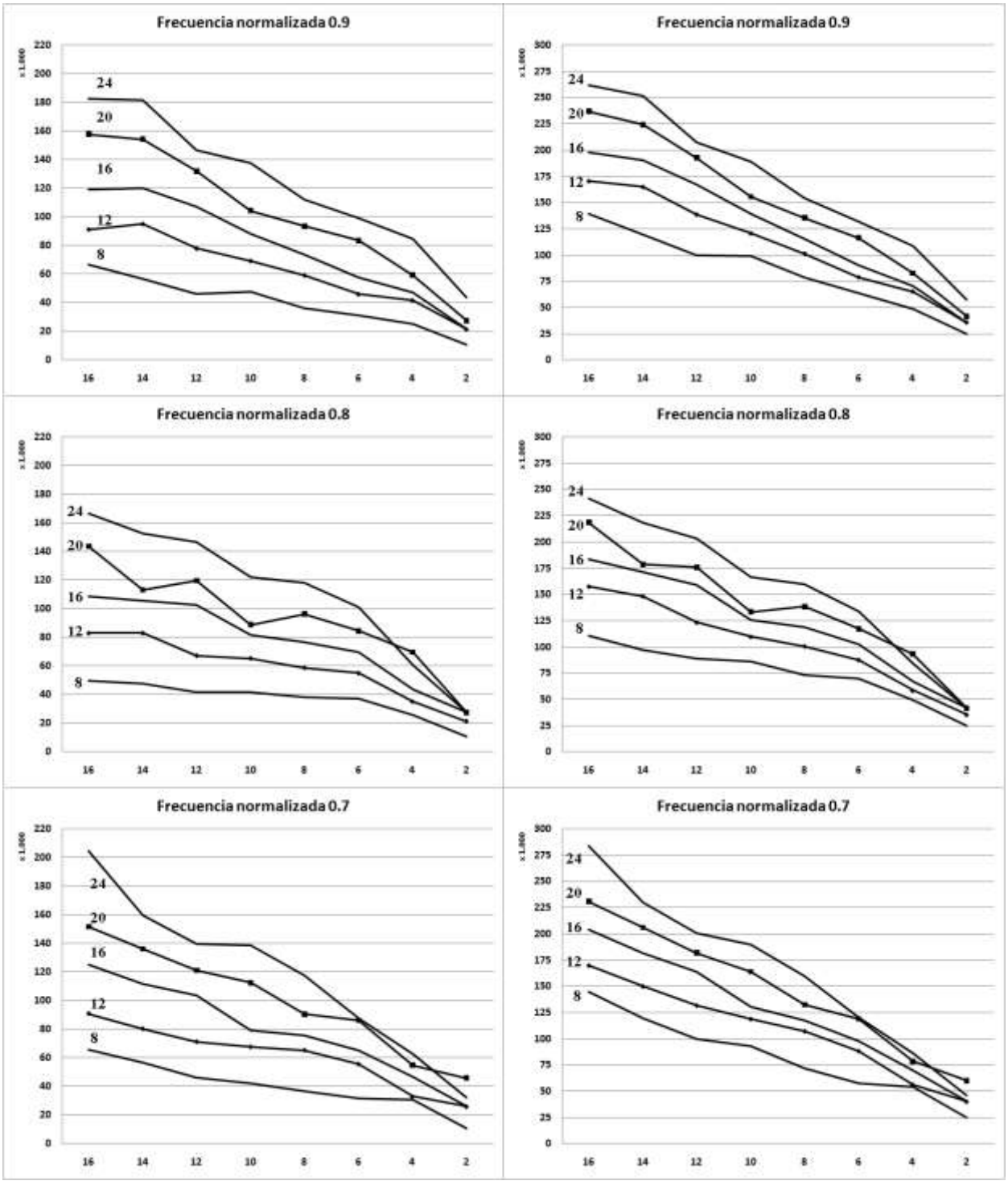


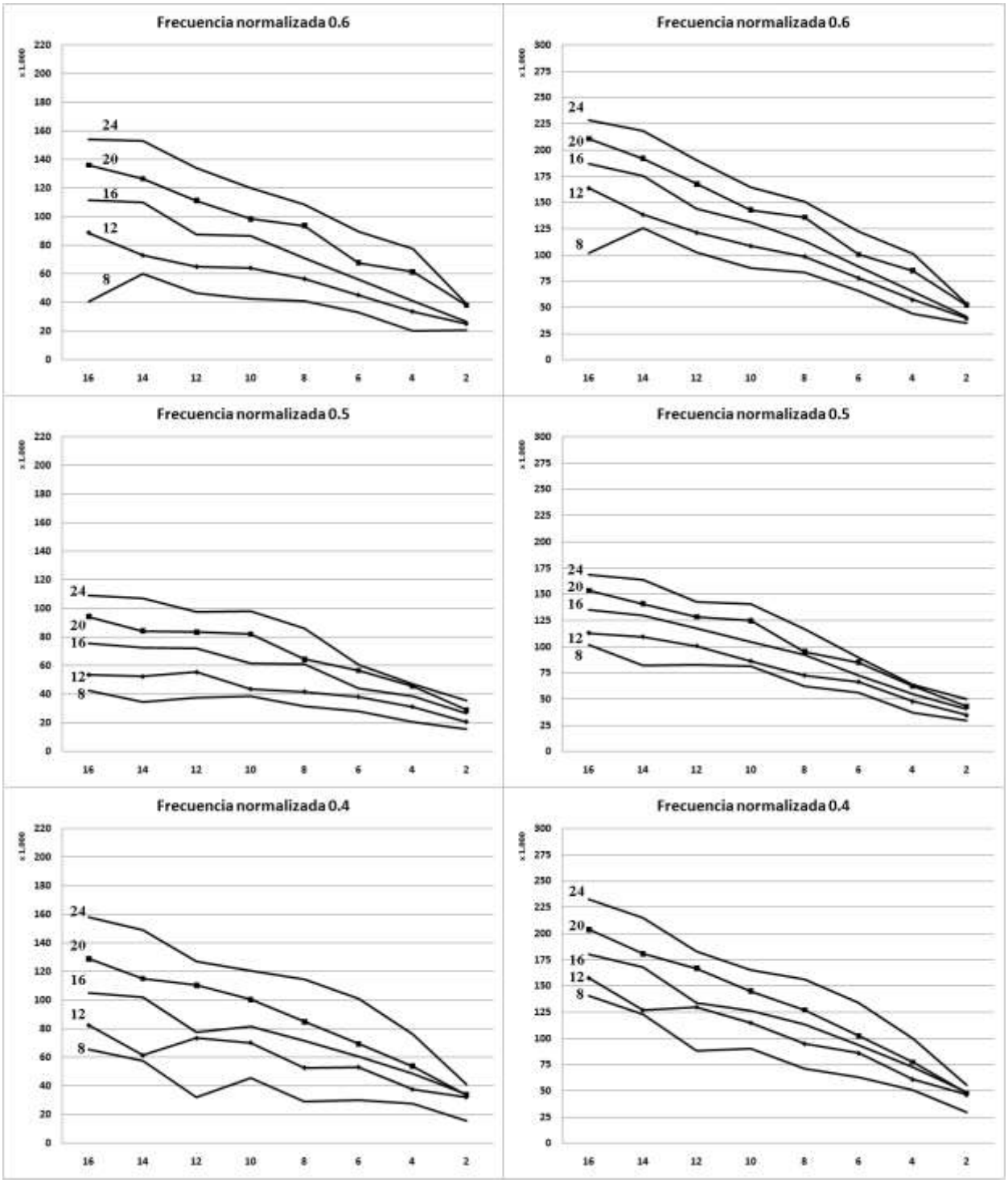
Figura 4.29 Porcentaje del Área aportada por el Comparador FIR DMR Paso alto (Spiral)

En las figuras 4.30, 4.31 y 4.32 se representan, para cada frecuencia normalizada de corte del filtro, los valores de áreas del conjunto formado por los dos Bloques Multiplicadores estándar (Bloque Spiral) y su Comparador y el área total de circuito para Filtros FIR DMR Paso alto calculados con precisión 8 a 24 bit y orden par comprendido entre 2 y 16.



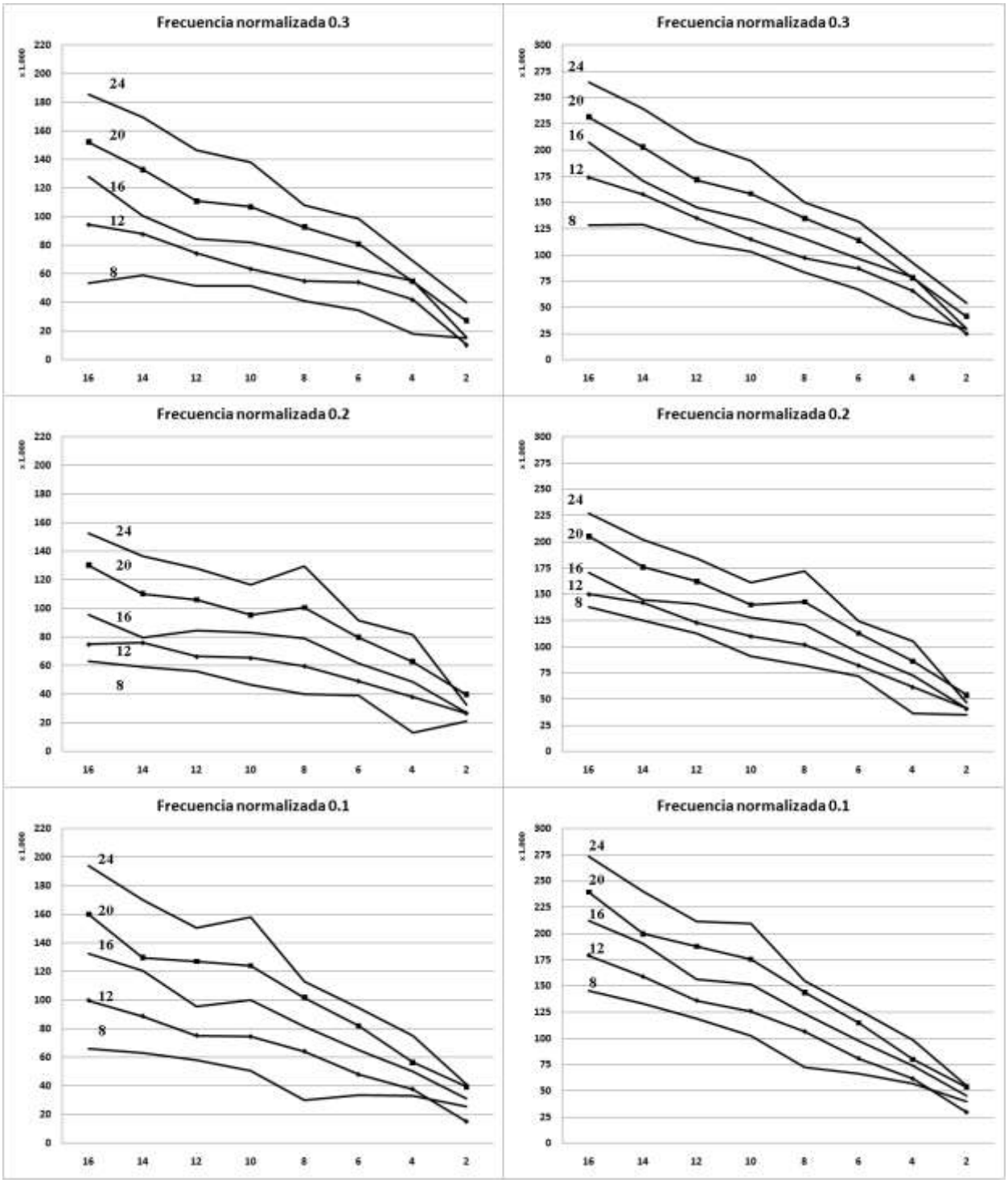
Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.30 Área Bloque Multiplicador/Comparador y Área total Filtro DMR Paso alto (1).



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro. La precisión de cálculo, se indica en las propias gráficas

Figura 4.31 Área Bloque Multiplicador/Comparador y Área total Filtro DMR Paso alto (2).



Nota: El área representada en el eje de ordenadas tiene unidades de Lambda^2. En el eje de abscisas se representa el orden del filtro La precisión de cálculo, se indica en las propias gráficas.

Figura 4.32 Área Bloque Multiplicador/Comparador y Área total Filtro DMR Paso alto (3).

El porcentaje de área aportado al total del circuito por el Bloque Multiplicador DMR varía desde una media de aproximadamente el 21% para cálculos con 8 bit de precisión a una media de más del 50% para precisión de 24 bit.

En la figura 4.33 se representa dicha variación según la precisión de cálculo utilizada.

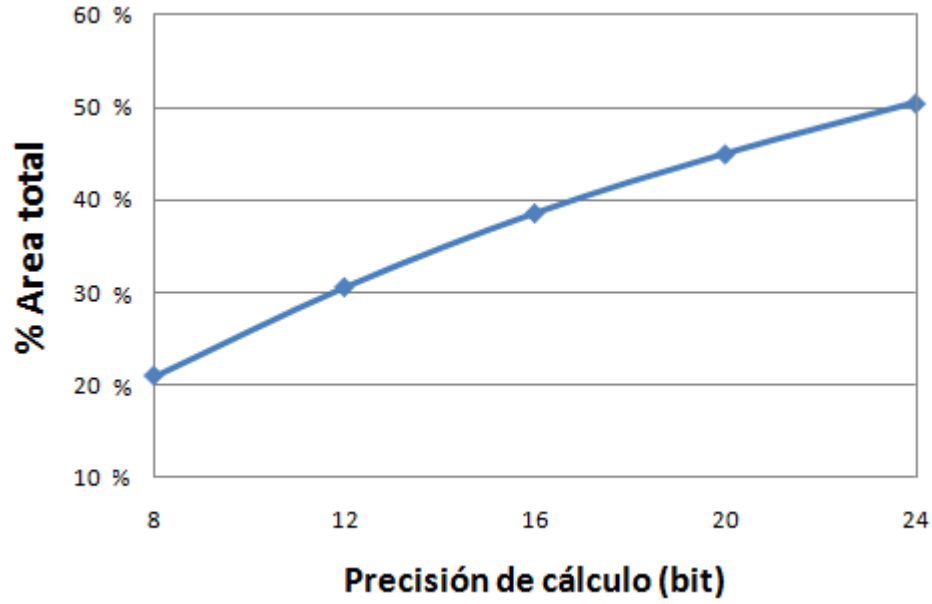


Figura 4.33 Porcentaje Área aportada por Bloque Multiplicador FIR DMR Paso bajo (Spiral)

4.4 Ahorro de superficie total en circuitos que implementan Filtros FIR DetectError.

Para determinar cuantitativamente el tanto por ciento de ahorro de área al implementar Filtros FIR Paso bajo y Paso alto DetectError que trabajan con precisión de cálculo de 8 a 24 bit, orden comprendido entre 2 y 16 y frecuencia normalizada de corte comprendida entre 0'1 y 0'9, se han utilizado las siguientes expresiones:

$$Filtro\ FIR\ Paso\ bajo$$

$$Ahorro\ Area\ (Wn,N)\% \ = \ 100 * \left(1 - \frac{Area\_DetectError\ Paso\ bajo(Wn,N)}{Area_{DMR} Paso\ bajo(Wn,N)}\right)$$

$$Filtro\ FIR\ Paso\ alto$$

$$Ahorro\ Area\ (Wn,N)\% \ = \ 100 * \left(1 - \frac{Area\_DetectError\ Paso\ alto(Wn,N)}{Area_{DMR} Paso\ alto(Wn,N)}\right)$$

Cuando el ahorro es un número positivo, significa que el Método DetectError es más eficiente que el Método de Replicación Dual Modular pues genera circuitos lógicos más pequeños.

Por el contrario, cuando el ahorro es un número negativo, significa que el Método DetectError es menos eficiente que el Método de Replicación Dual Modular y se generan circuitos lógicos más grandes.

Los valores de ahorro de superficie total obtenidos para todos los Filtros FIR DetectError Paso bajo y Paso alto con precisión de cálculo 8 a 24 bit y orden de filtro 2 a 16 estudiados, se recogen, respectivamente en las tablas 4.83 a 4.87 y 4.88 a 4.92

En las figuras 4.34 y 4.35 se indica, respectivamente, la distribución estadística que representa el ahorro de área para cada una de las precisiones de cálculo de los Filtros FIR Paso bajo y Paso alto estudiados.

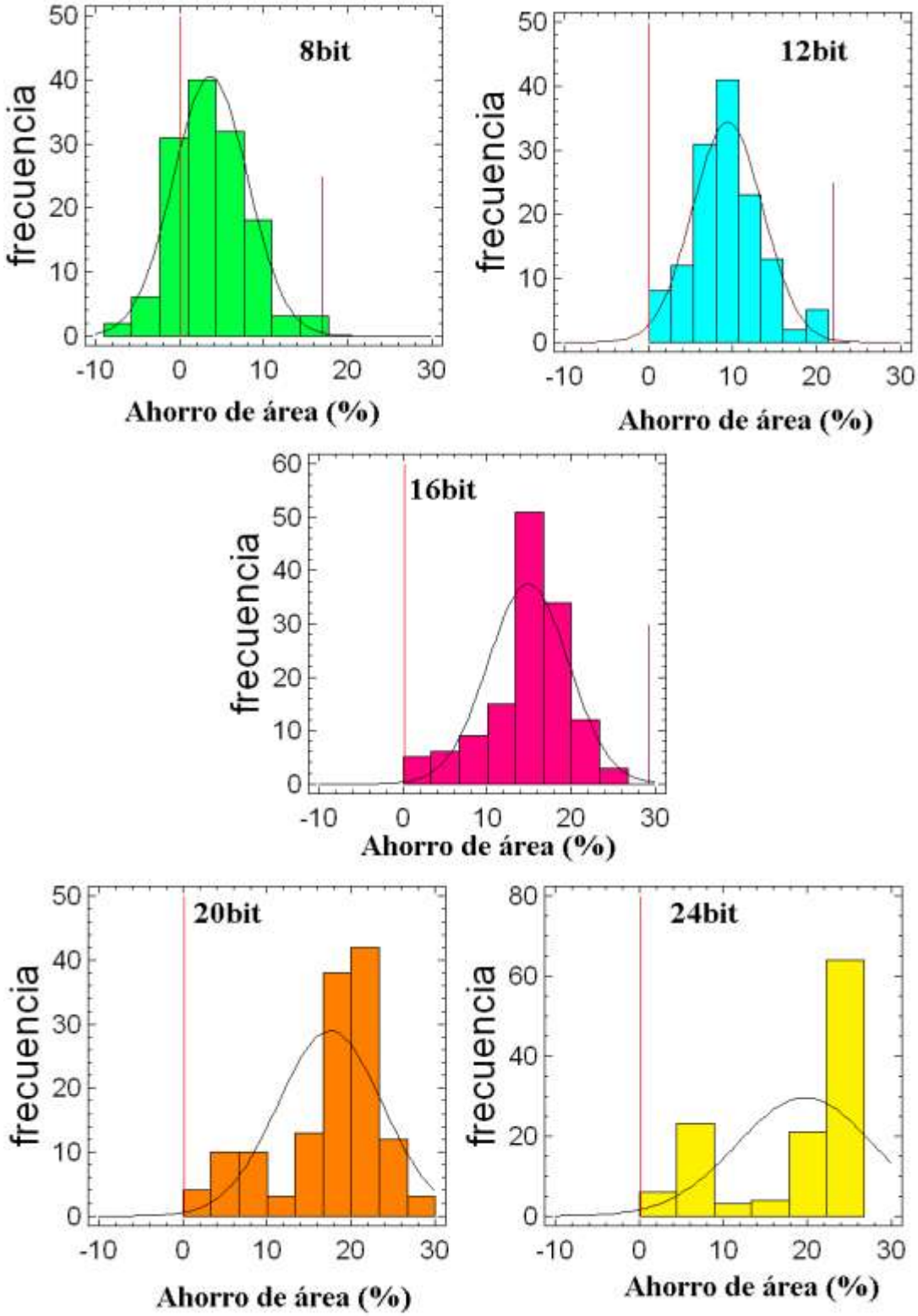


Figura 4.34 Distribución estadística del ahorro de área Filtros FIR Paso bajo DetectError.

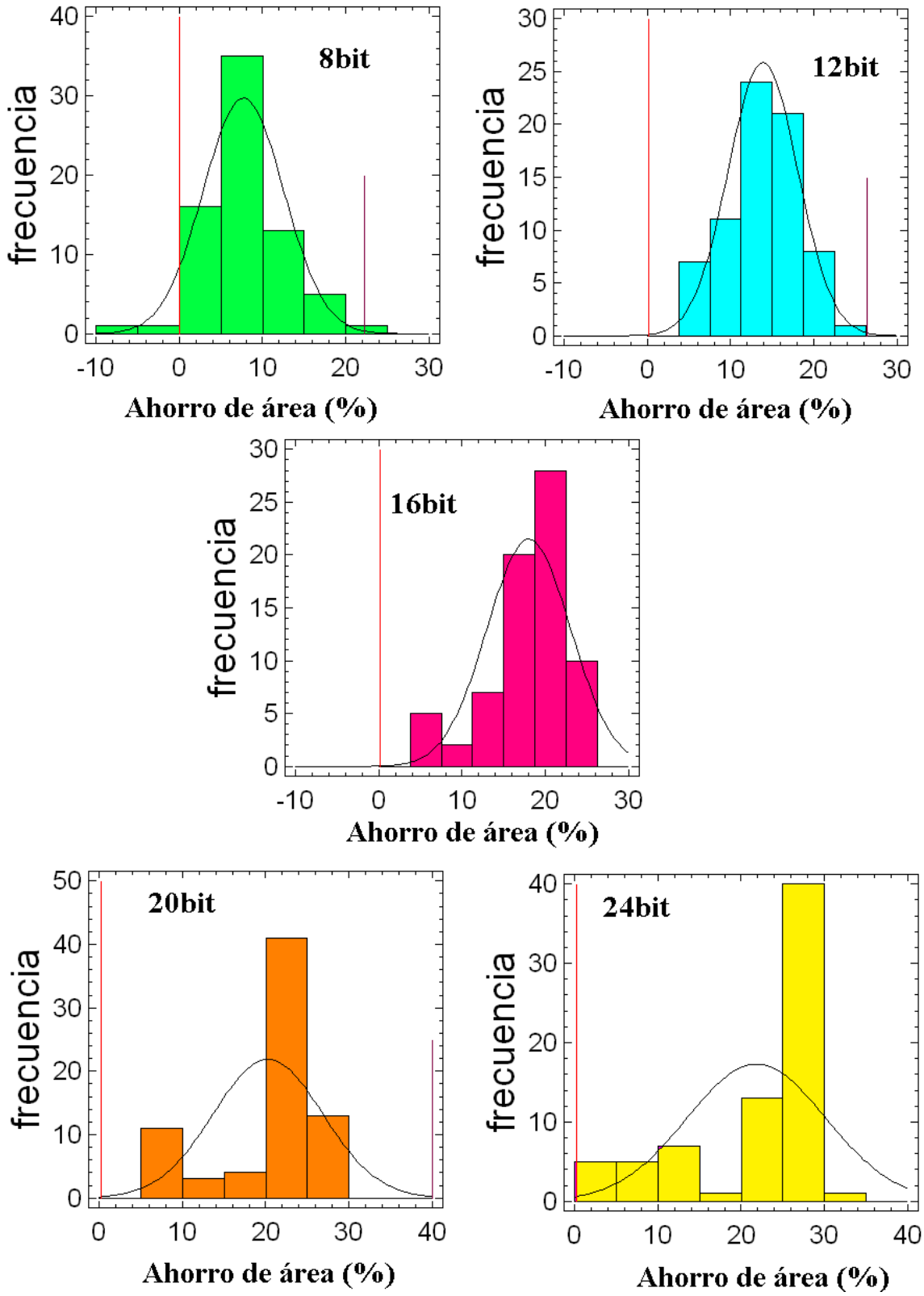


Figura 4.35 Distribución estadística del ahorro de área Filtros FIR Paso alto DetectError.

En las tablas 4.93 y 4.94, se recoge, respectivamente, el valor medio de Ahorro de Área, la desviación estándar del Ahorro de Área, y los valores máximos y mínimos de ahorro de área conseguidos al utilizar Filtros FIR DetectError Paso bajo y Paso alto, según la precisión de cálculo empleada.

Tabla 4.93 Valor medio de Ahorro de Área y desviación estándar. Filtro FIR Paso bajo

Precisión de Cálculo	Ahorro Medio (%)	Desviación estándar (%)	Ahorro Máximo (%)	Ahorro Mínimo (%)
8 bit	3.68	4.43	15.75	-7.74
12 bit	9.39	4.17	21.07	1.31
16 bit	14.88	4.79	24.78	1.77
20 bit	17.53	6.20	28.00	2.09
24 bit	19.69	8.09	30.94	2.10

Tabla 4.94 Valor medio de Ahorro de Área y desviación estándar. Filtro FIR Paso alto

Precisión de Cálculo	Ahorro Medio (%)	Desviación estándar (%)	Ahorro Máximo (%)	Ahorro Mínimo (%)
8 bit	7.72	4.83	22.46	-7.56
12 bit	13.87	4.16	24.67	4.91
16 bit	17.99	5.01	26.08	4.35
20 bit	20.27	6.57	29.74	5.07
24 bit	21.83	8.30	31.11	4.14

A la vista de los datos contenidos en dichas tablas, podemos extraer las siguientes conclusiones:

- Salvo para un 17.8% de Filtros FIR DetectError Paso bajo de 8 bit de precisión y un 2.8% Filtros FIR DetectError Paso alto de 8 bit de precisión, en los que el Método DetectError ofrece un valor inferior de área al Método DMR, en todos los demás casos, el Método DetectError proporciona siempre ahorros positivos.
- El ahorro de área proporcionado por el Método DetectError, para ambos tipos de Filtrado es mayor cuanto mayor es la precisión de cálculo de diseño del Filtro FIR, llegando a superar valores del 30% para Filtros de 24 bit de precisión de cálculo.

  En efecto, a mayor necesidad de precisión, el Bloque Multiplicador presentará una estructura más compleja, que deberá ser duplicada completamente en el diseño DMR pero que tan solo contará con algunos nodos replicados en el diseño DetectError.

- El ahorro de área proporcionado por el Método DetectError para Filtros FIR Paso bajo y Paso alto de 20 y 24 bit de precisión presenta Desviaciones estándar más elevadas que en los Filtros de precisión de cálculo 8, 12 y 16 bit.
  Ello es debido a la importante contribución de los Bloques Multiplicadores que tienen nodos replicados, y que dado el número elevado de bit de precisión requerida, son mucho más complejos.

  Como puede comprobarse en las tablas 4.95 y 4.96, donde se representa, respectivamente, por separado, el ahorro de área de Filtros FIR Paso bajo y Paso alto con Bloques Multiplicadores en los que no ha sido necesario replicar ningún nodo y Filtros FIR Paso bajo y Paso alto con nodos replicados, el ahorro medio es prácticamente el doble en la primera de las situaciones.

Tabla 4.95 Ahorro Área Filtros FIR Paso bajo con Precisión cálculo 20 y 24 bit

	Bloques Multiplicadores Sin replicados		Bloques Multiplicadores Con replicados	
Precisión De cálculo	**Ahorro medio Área**	**Desviación estándar**	**Ahorro medio Área**	**Desviación estándar**
20 bit	20,98	2,65	11,25	5,87
24 bit	24,47	2,31	12,29	8,28

Tabla 4.96 Ahorro Área Filtros FIR Paso alto con Precisión cálculo 20 y 24 bit

	Bloques Multiplicadores Sin replicados		Bloques Multiplicadores Con replicados	
Precisión De cálculo	**Ahorro medio Área**	**Desviación estándar**	**Ahorro medio Área**	**Desviación estándar**
20 bit	25,57	2,06	14,26	7,68
24 bit	28,04	1,55	14,77	8,79

En las figuras 4.36 y 4.37, se visualiza, respectivamente, la diferencia existente ente las distribuciones de frecuencia de Filtros FIR DetectError Paso bajo y Paso alto de 20 y 24 bit de precisión de cálculo que no tienen nodos replicados (**nr20** y **nr24**) y las de Filtros FIR DetectError Paso bajo y Paso alto de 20 y 24 bit de precisión de cálculo que si tienen replicados en sus nodos (**r20** y **r24**).

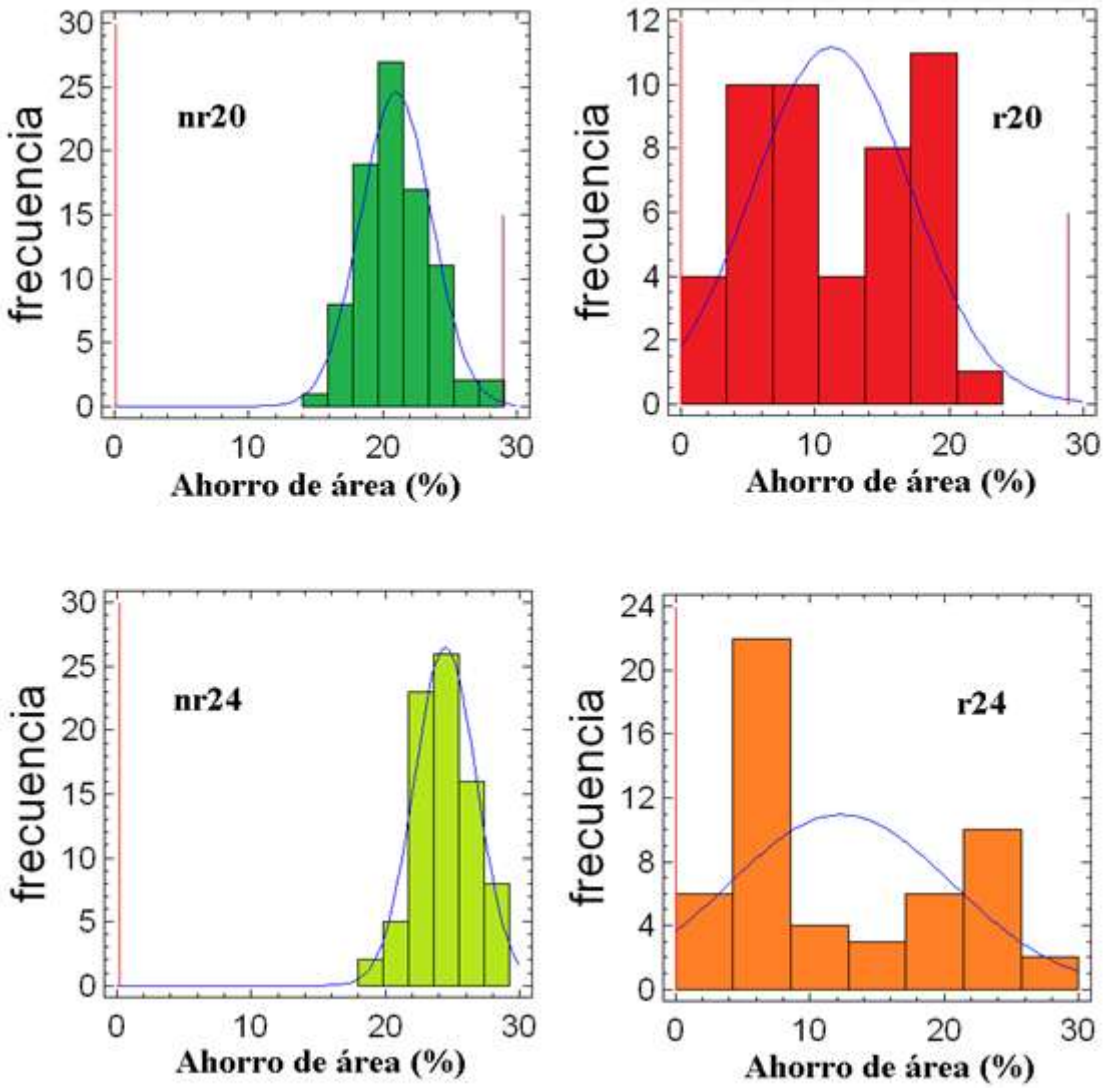


Figura 4.36 Distribución frecuencia Filtros FIR DetectError Paso bajo 20 y 24 bit.

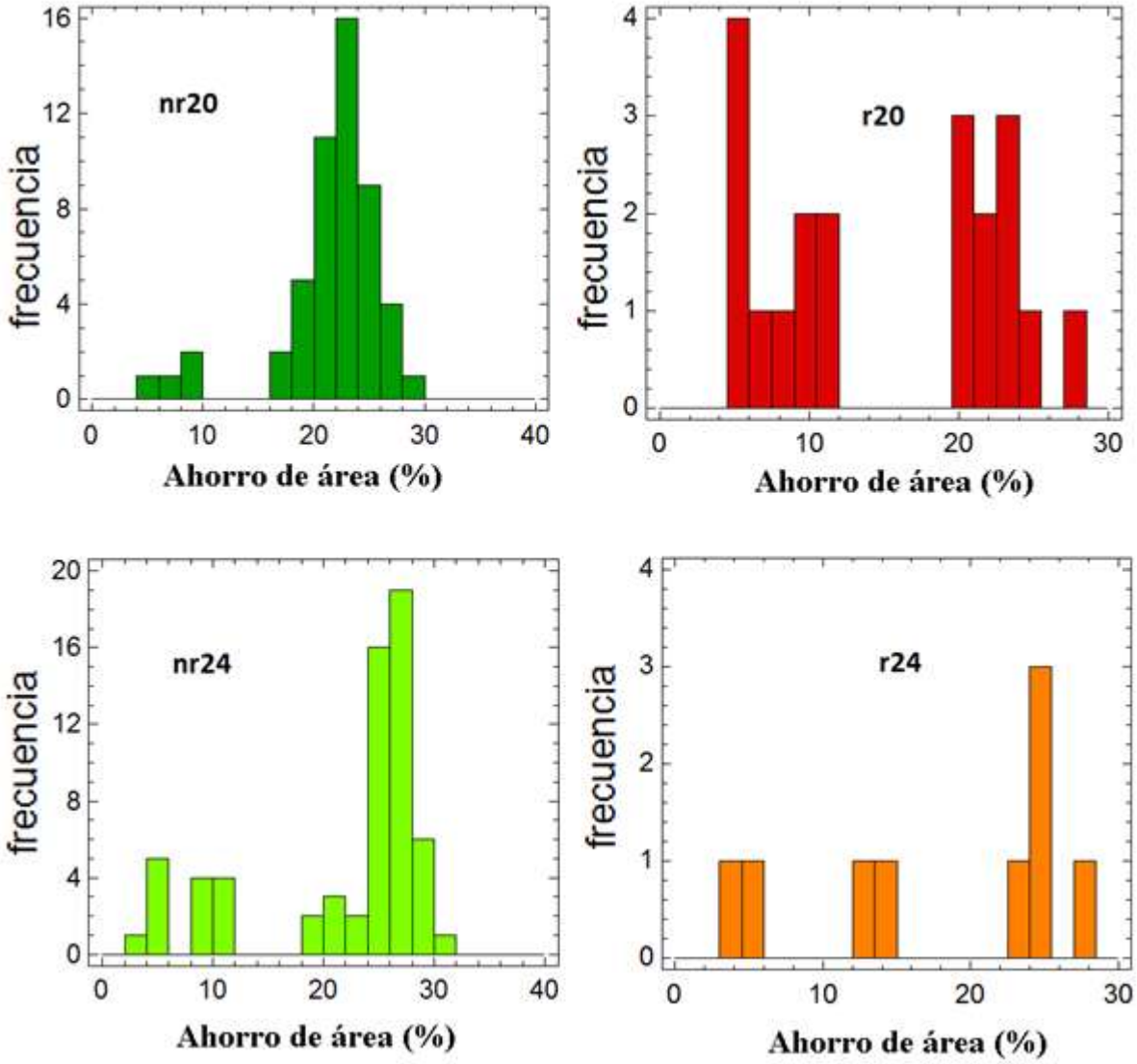


Figura 4.37 Distribución frecuencia Filtros FIR DetectError Paso alto 20 y 24 bit.

Capítulo 5 - Simulación digital de Soft Errores en BM DetectError.

Para seguir el procedimiento habitual establecido por la IEEE y confirmar la efectividad de la Técnica de Replicación Parcial de Nodos, se procederá a comprobar, mediante simulación digital, la detección de Soft Errors en la estructura de un Bloque Multiplicador DetectError

Para ello se ha seguido el esquema lógico de trabajo indicado en la figura 5.1.

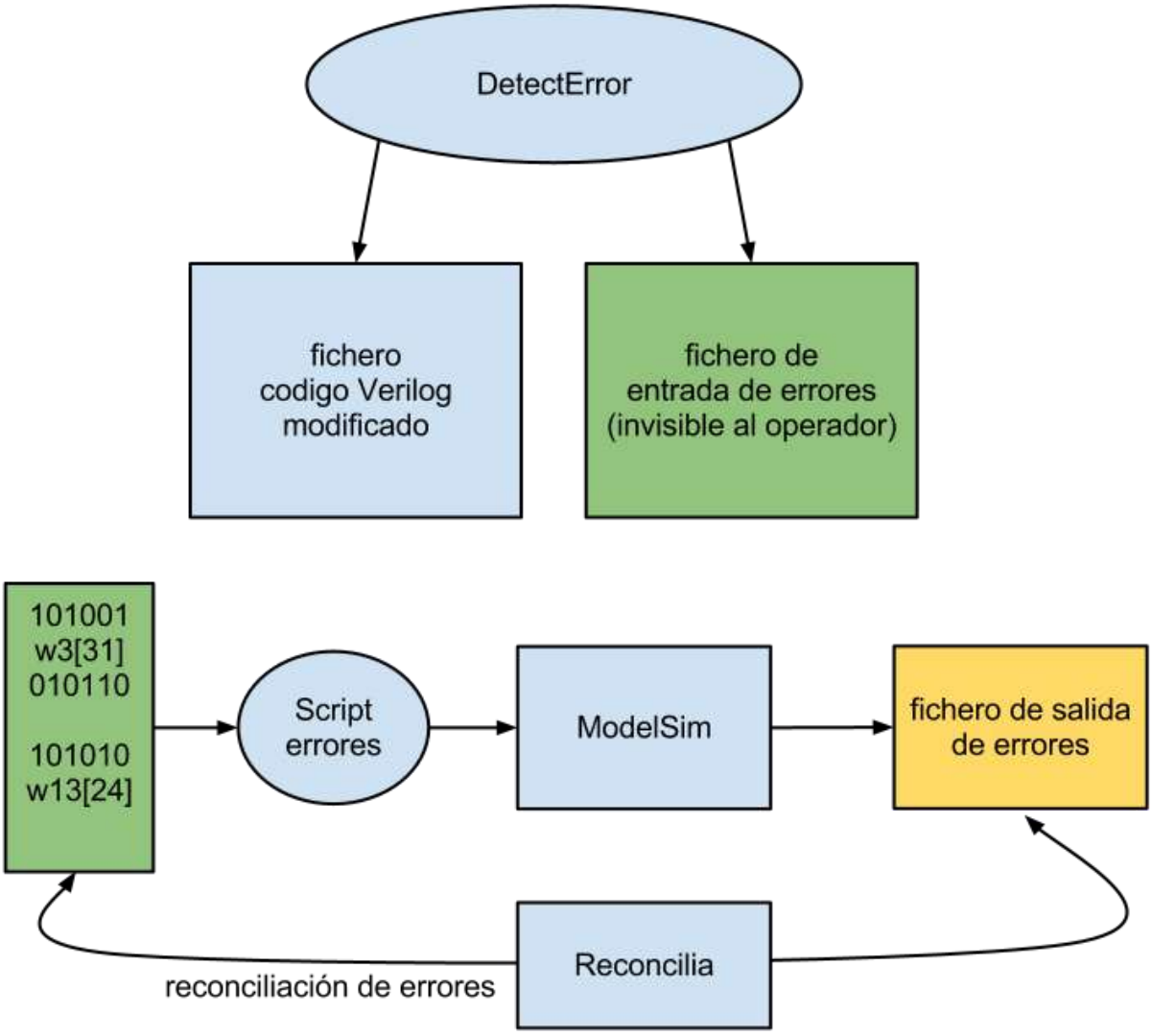


Figura 5.1 Esquema de trabajo de la Simulación.

Fichero de errores.

El programa DetectError, tras llevar a cabo la replicación de los nodos necesarios para evitar la compensación de errores, genera de forma aleatoria valores de entrada al Bloque Multiplicador que a su vez pueden estar afectados en un 80% de los casos por un Soft Error. Cuando se produce dicho error el bit de destino es elegido también de forma totalmente aleatoria.

Esos datos se guardan de forma secuencial en un archivo tipo texto con extensión .dat, al que llamaremos Fichero de entrada de errores.

En primer lugar se archiva el valor de la entrada en binario con 32 bit y en segundo lugar se indica el nodo y el bit que ha sufrido Soft Error. En caso de no producirse Soft Error el dato guardado es una línea en blanco.

En la figura 5.2 se presentan, a modo de ejemplo, las veinte primeras líneas del fichero de errores generado por DetectError al procesar los datos correspondientes a un Filtro FIR Paso bajo de orden 3, frecuencia normalizada de corte 0’1 y 8 bit de precisión.

Como puede comprobarse en los sucesos 4 y 9 no se ha producido Soft Error,

```
1  00010000010100001110110100011101
   w32[0]
2  00110000010010001111111010001110
   w32[39]
3  00011101100101010001001001001101
   w3[23]
4  11011011110111110001100100110000

5  11000111011001010100000100000101
   w8[35]
6  10001001001101101000110000010110
   w32[14]
7  01000000111111000100100101010101
   w8[20]
8  10100011010010000101011010011001
   w29[6]
9  10000111011001100111010001100110

10 01110001011000111010110010001110
   w8[22]
```

Figura 5.2 Ejemplo de fichero de entrada de errores generado por DetectError.

El proceso de generación de errores es invisible a los ojos del usuario para que no se encuentre mediatizado en la posterior operación de reconocimiento de errores

Técnica de Simulación.

La simulación del comportamiento del Bloque Multiplicador se lleva a cabo mediante el concurso de la herramienta ModelSim [14].

Script de errores.

La inserción de errores en el Bloque Multiplicador simulado digitalmente con la herramienta ModelSim, se lleva a cabo de forma automática mediante la implementación de un Script en lenguaje Tcl [15], que se encarga de ir leyendo los datos contenidos en el Fichero de entrada de errores.

El código Verilog creado con DetectError al simularse en ModelSim propicia la generación de un fichero de salida de errores tipo texto en el que se recogen los datos característicos de salida del Bloque Multiplicador: Registro Check y todos sus nodos intermedios y finales, en dos momentos del proceso:

1) Cuando se introduce el valor de la entrada X.
2) Cuando se fuerza el error en el nodo y bit indicado en el fichero de entrada de errores.

Este fichero es también invisible a los ojos del usuario de la simulación.

Reconciliación de errores.

La última etapa de esta simulación es la reconciliación de errores. Para ello se aprovechan los datos contenidos en el fichero de salida de errores, comprobando si los errores presentes en el fichero de entrada de errores inicial se corresponden o no con los detectados por el Registro Check y guardados en el fichero de salida de la simulación generado por ModelSim.

5.1 Respuestas del Registro Check ante un Soft Error.

Desde el punto de vista teórico, ante un Soft Error, uno o más nodos del Bloque Multiplicador generan un valor erróneo de salida que, al no compensarse durante el resto de operaciones, da lugar a un resultado erróneo en el valor del sumatorio final. En esa situación, el Registro Check confirma la aparición del Soft Error proporcionando un valor de salida distinto de cero.

Esta respuesta positiva de Soft Error no lleva asociada inequívocamente la existencia de error en los valores de las salidas del Bloque Multiplicador.

En la figura 5.3 se recogen las posibles situaciones generadas en cada ciclo de la simulación, tras la entrada de una señal que puede estar afectada o no por un Soft Error.

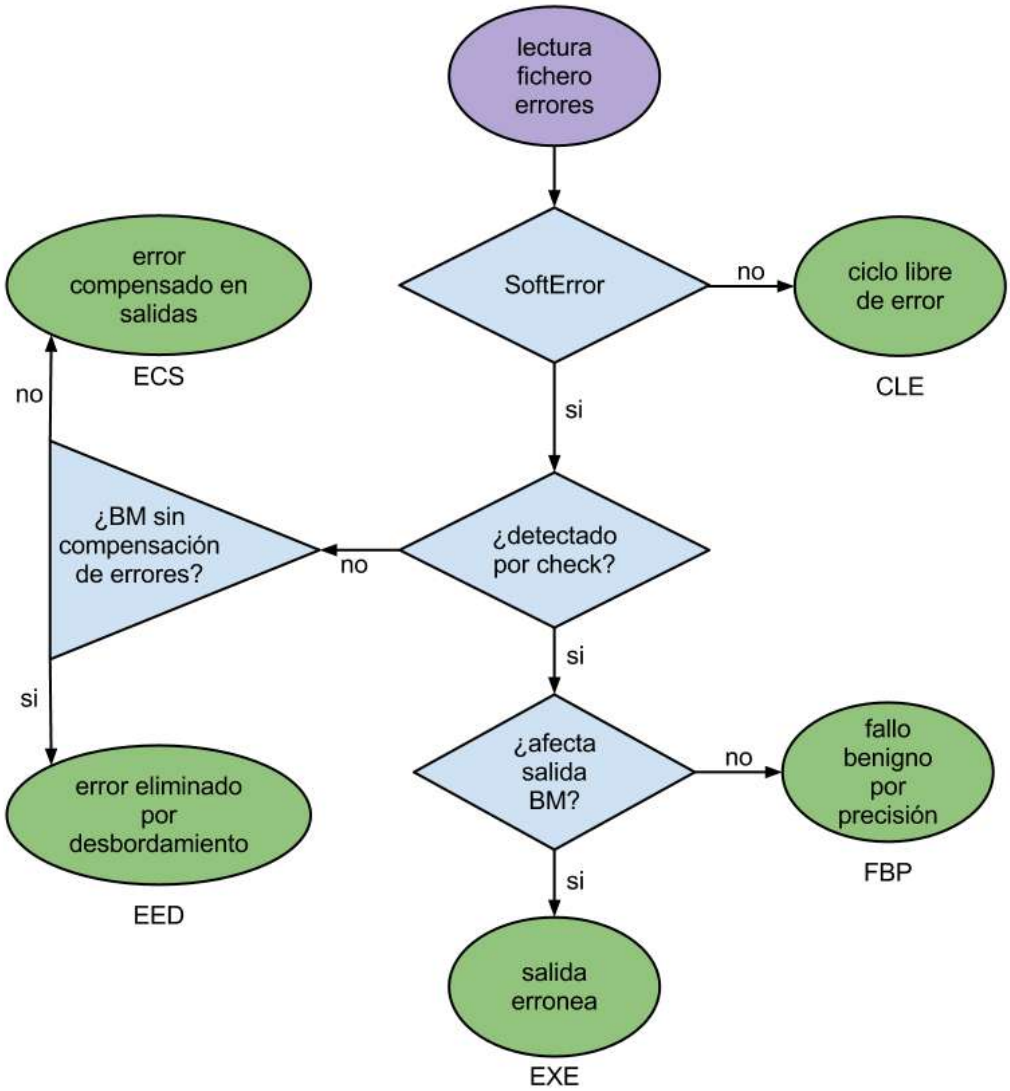


Figura 5.3 Posibles resultados de cada ciclo de simulación.

Para aclarar todas esas situaciones, se utilizarán, en todos los casos, los resultados de la simulación digital del grafo del Bloque Multiplicador recogido en la figura 5.4, obtenido mediante el concurso de DetectError. Corresponde al Bloque Multiplicador de un Filtro FIR Paso bajo de Orden 3, frecuencia normalizada de corte 0'1 y 8 bit de precisión de cálculo.

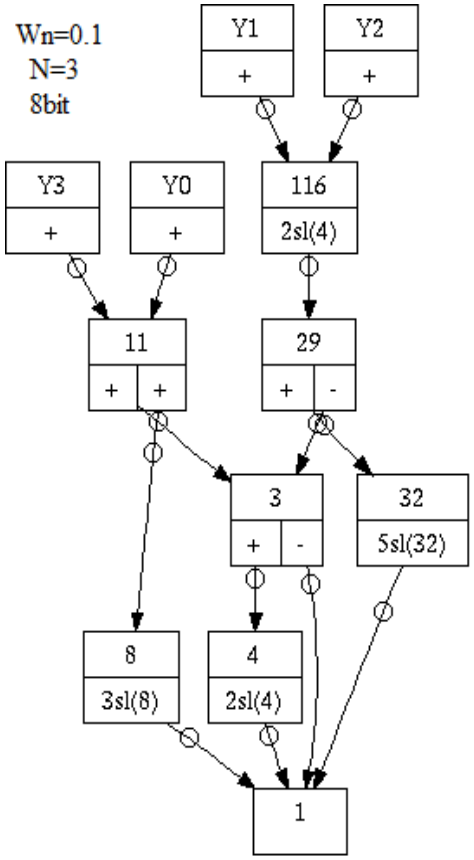


Figura 5.4 Grafo del Bloque Multiplicador usado como ejemplo en la simulación.

5.1.1 Ciclo de simulación sin Soft Error: CLE.

Una vez que se produce la entrada de un valor X en el nodo w1 del Bloque Multiplicador, tienen lugar todas las operaciones matemáticas descritas en el grafo representativo de dicho Bloque, consistentes en la realización de sumas, restas y desplazamiento de bit.

Como resultado, todos y cada uno de los nodos contienen su valor correspondiente. En la figura 5.5 se representa el contenido de todos los nodos del Bloque Multiplicador del ejemplo elegido.

Nodo bit	X	w1	w4	w3	w8	w11	w32	w29	w116	wckeck	ckeck
39	s	s	S	s	s	s	s	s	s	s	s
38	s	s	S	s	s	s	s	s	s	1	0
37	s	s	S	s	s	s	s	s	0	0	0
36	s	s	S	s	s	s	s	s	1	0	0
35	s	s	s	s	s	s	1	0	1	0	0
34	s	s	s	s	s	0	0	1	1	0	0
33	s	s	s	s	1	0	0	1	0	0	0
32	s	s	1	0	0	1	0	1	1	0	0
31	s	s	0	1	0	1	0	0	0	0	0
30	1	1	0	1	0	1	0	1	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
Operación			1<<2	4 - 1	1<<3	8+3	1<<5	32 - 3	29<<2	116<<1+11<<1+2	

Figura 5.5 Representación valores en nodos del Bloque Multiplicador.

En color verde se indican los bits asignados a cero por haberse realizado en el nodo un desplazamiento. En color azul los bit procedentes de la suma o resta de los dos nodos previos.

En esas condiciones, el registro wcheck, procede a sumar todos los nodos que alimentan las salidas.

En nuestro ejemplo:

wcheck = w11 + w11 + w116 + w116 + (w1 << 1)

Y el Registro Check está en condiciones de anular los 32 bit más significativos del registro wcheck con el valor de la entrada X.

El Registro Check presenta todos sus bits a cero y por tanto se confirma que no se ha producido Soft Error durante el ciclo simulado.

5.1.2 Soft Errors de Desbordamiento: Tipos.

Cuando un Soft Error se produce en cualquier nodo, N_SoftError, que está conectado con un nodo superior en el que se produce desplazamiento de bits, N_Desplaz, puede afectar a las salidas, o no afectarlas, dependiendo de la posición del bit afectado en el nodo N_SoftError y del número de desplazamientos que contabilice el nodo N_Desplaz.

En efecto, tal y como se refleja en la figura 5.6, el Soft Error se detectará, como consecuencia de afectar a una o más salidas del Bloque Multiplicador, cuando en el nodo en el que hay desplazamiento, el bit destinatario exista, es decir cuando el nodo en cuestión tenga un número de bits mayor o igual que:

Número bits N_Desplaz >= Bit N_SoftError + nº de desplazamientos.

Por el contrario, tal y como queda reflejado en la figura 5.7, no afectará a ninguna salida del Bloque Multiplicador, no siendo detectado por el Registro Check y por tanto no siendo cuantificado como Soft Error, cuando:

Número bits N_Desplaz < Bit N_SoftError + nº de desplazamientos.

Ya que en esta segunda situación se produce un desbordamiento de bits y el error debería colocarse en un bit que no existe. Esta situación se corresponde con el denominado Fallo Benigno, también llamado No Error Aparente

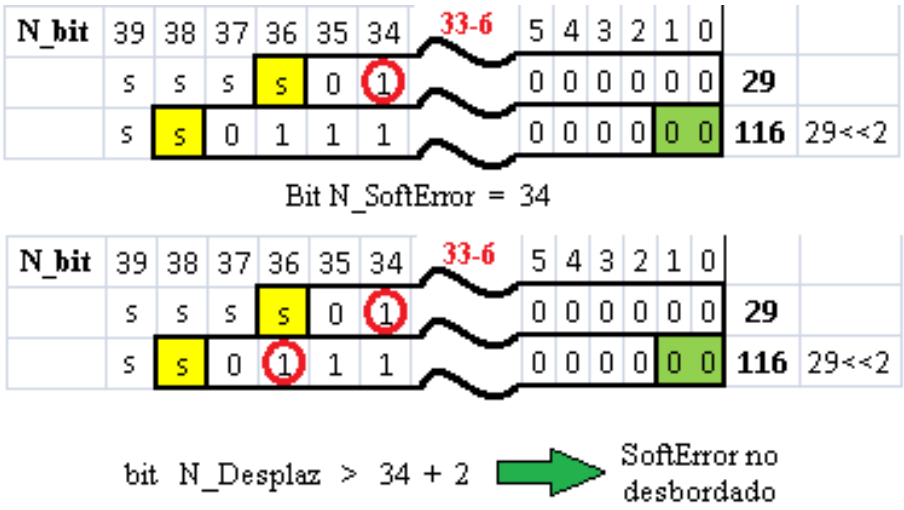


Figura 5.6 Soft Error no desbordado.

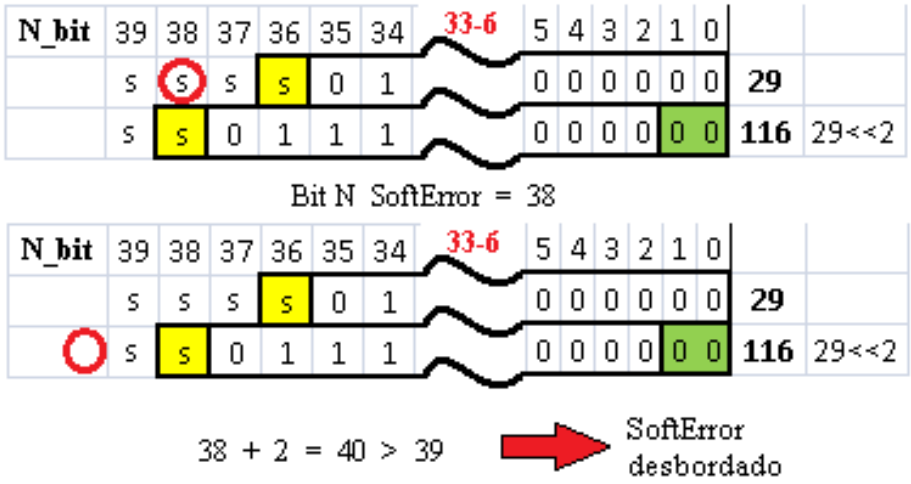


Figura 5.7 Soft Error desbordado.

5.1.3 Soft Errors detectados por el Registro Check DetectError.

Para facilitar las operaciones de suma y resta que tienen lugar en los nodos de un Bloque Multiplicador, los coeficientes originales proporcionados por MATLAB se transforman a números enteros.

Una vez obtenidas las salidas del Bloque Multiplicador, y antes de ser utilizadas en los cálculos posteriores que se efectúan en el Filtro FIR, deben ser de nuevo transformadas, eliminando para ello el número de fractional bit empleado en el menú de cálculo de Spiral, y que fue seleccionado para conseguir la precisión deseada.

Esta operación se realiza de forma muy sencilla, mediante un simple desplazamiento de bits hacia la derecha.

Dependiendo del tipo de nodo afectado y del valor del bit eliminado se pueden producir dos situaciones bien distintas. En una de ellas se verá afectado el valor de salida dando lugar a un ERROR DE SALIDA (EXE), en la otra, al no producirse dicha afectación, el Soft Error origina un FALLO BENIGNO POR PRECISON DE CALCULO (FBP).

A continuación se explicarán ambas situaciones, con sendos ejemplos aclaratorios.

Fallo Benigno Eliminado por precisión de Cálculo: FBP.

Situación: Soft Error en bit 7 del nodo w116.

El registro wcheck efectúa la operación

wcheck = w11 + w11 + w116 + w116 + (w1 << 1)

Como el nodo w116 interviene en dos ocasiones, el Soft Error se verá multiplicado por un valor de dos.

El registro wcheck contendrá por tanto un valor erróneo

Si el bit afectado en w116 ha pasado del valor “1” al “0”, el registro wcheck tendrá un defecto de 2^8 .Por el contrario, si el bit afectado en w116 hubiese pasado del valor “0” al “1”, el registro wcheck tendría un exceso de 2^8.

Al realizarse en el Registro Check la resta entre el valor del registro wcheck y el valor de entrada, mientras en el primer caso tendremos un resultado negativo, que afectará por tanto al bit 8 y superiores, en la segunda situación nos encontraremos con un exceso de 2^8 pero no se verá afectado ningún bit por encima de la posición 8.

En ambos casos, el Registro Check detectará la existencia de Soft Error pero como éste ha ocurrido en el bit 7 del nodo w116, situado por debajo del límite de precisión de cálculo, las salidas Y1 e Y2 del Bloque Multiplicador proporcionarán un valor correcto y los datos obtenidos por el Filtro FIR serán totalmente correctos.

Error de Salida: EEP.

Situación: Soft Error en bit 12 del nodo w116.

El Registro Check detectará la existencia de Soft Error y como ocurre en el bit 12 del nodo w116, situado dentro del límite de precisión de cálculo, las salidas Y1 e Y2 del Bloque Multiplicador serán incorrectas. Los datos obtenidos por el Filtro FIR serán erróneos. Se ha producido un Error de Salida.

5.2 Simulación Digital del Comportamiento de Bloques Multiplicadores estándar y DetectError frente a Soft Errors.

En este apartado se comparará mediante simulación digital, el comportamiento ante Soft Errors de un Bloque Multiplicador DetectError con replicación de nodos y equipado con Registro Check y un Bloque Multiplicador estándar Spiral.

Para realizar la simulación insertando los valores de entrada y errores contenidos en el fichero de entrada de errores generado por DetectError, se utiliza un Script construido en lenguaje Tcl [15], cuyo diagrama de flujo responde al esquema recogido en la figura 5.8.

En el anexo A-8 se encuentra el código fuente de este Script.

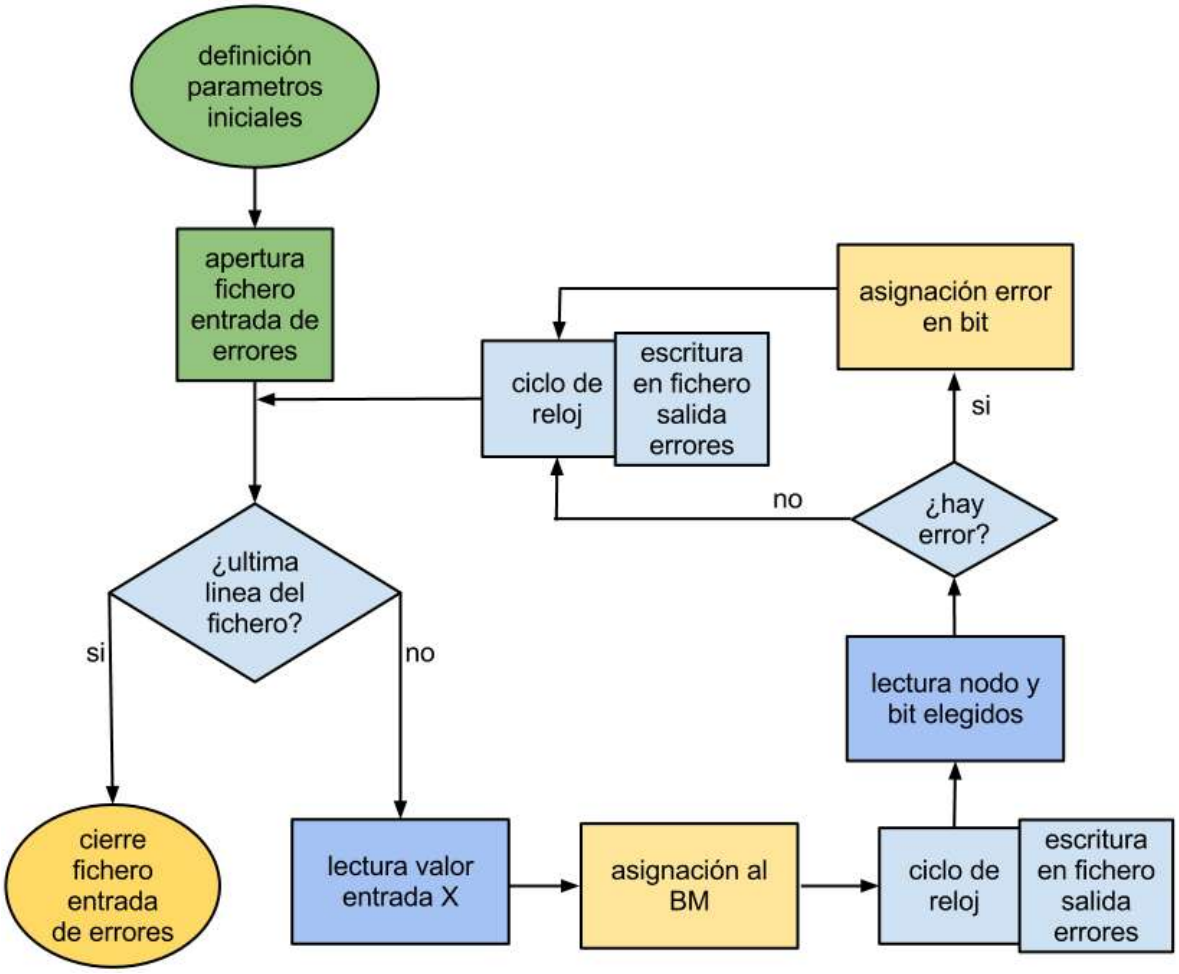


Figura 5.8 Diagrama de Flujo del Script de inserción de errores.

El objetivo es confirmar la superioridad de DetectError como método para detección de Soft Errors. Para ello se ha procedido a simular el comportamiento de un Bloque Multiplicador estándar

y un Bloque Multiplicador DetectError instalados en tres Filtros FIR Paso bajo cuyas características y cuyos grafos se representan en las figuras 5.9, 5.10, 5.11 .

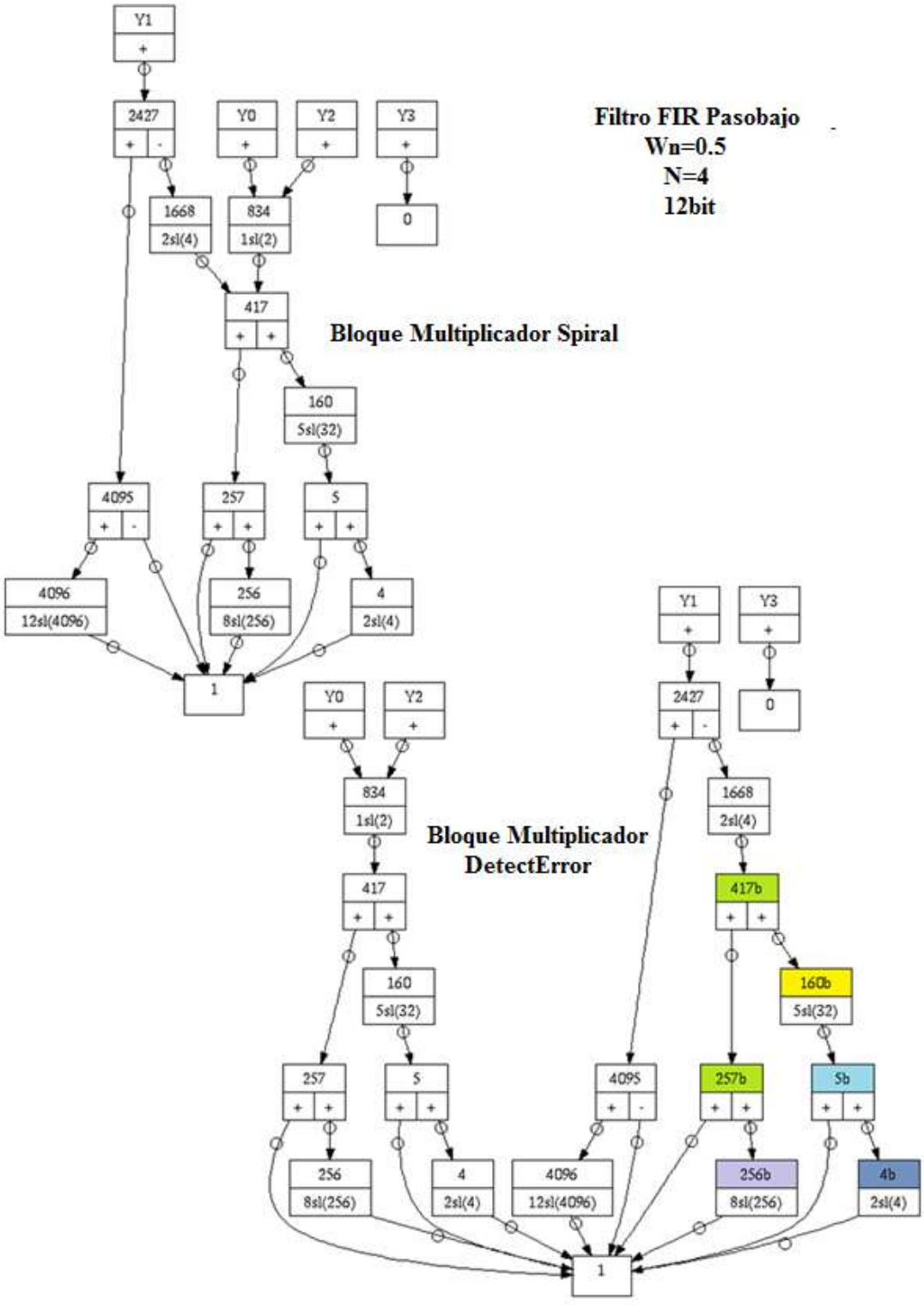


Figura 5.9 Bloques Multiplicadores Spiral y DetectError Paso bajo Orden 4.

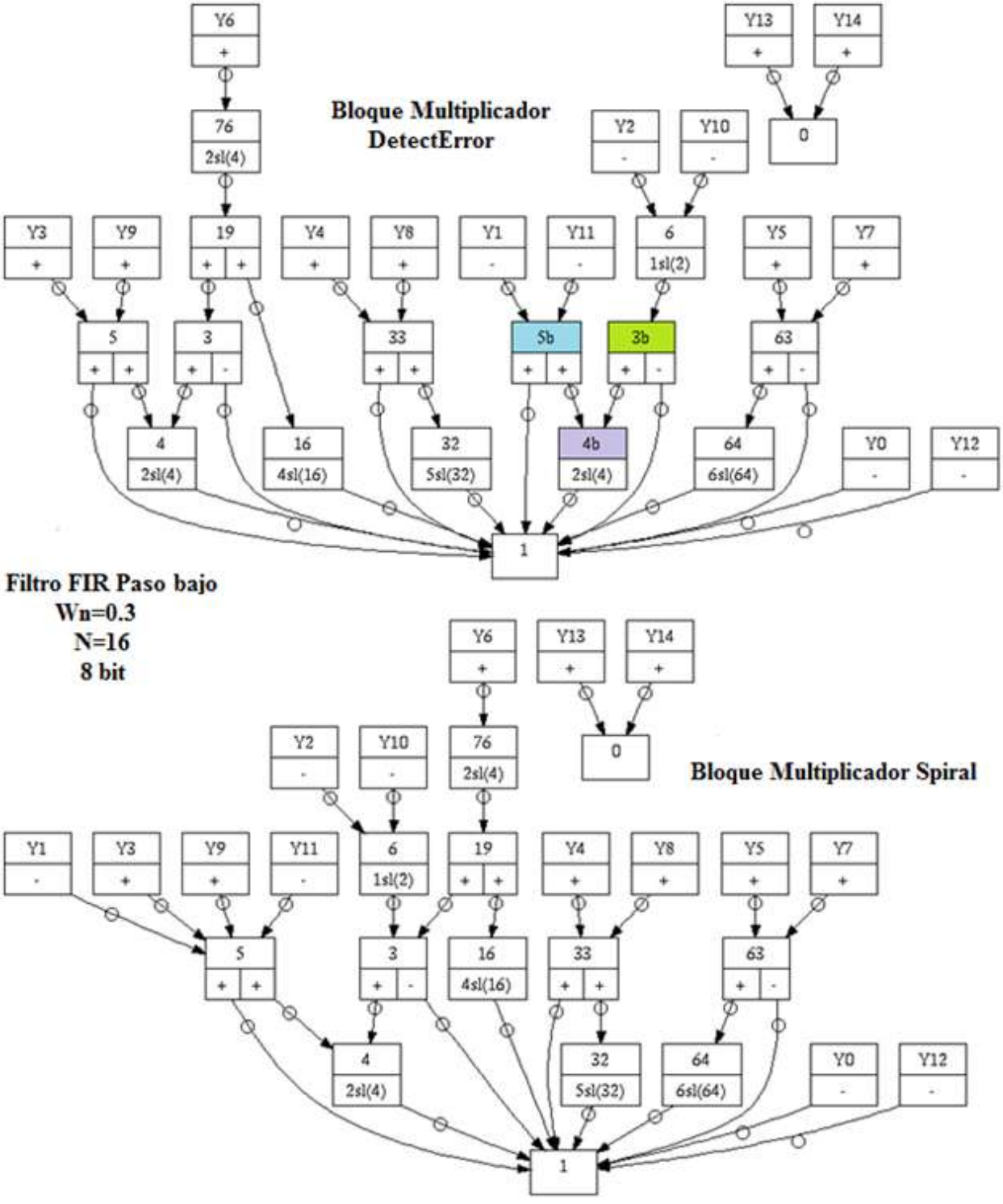


Figura 5.10 Bloques Multiplicadores Spiral y DetectError Paso bajo Orden 16.

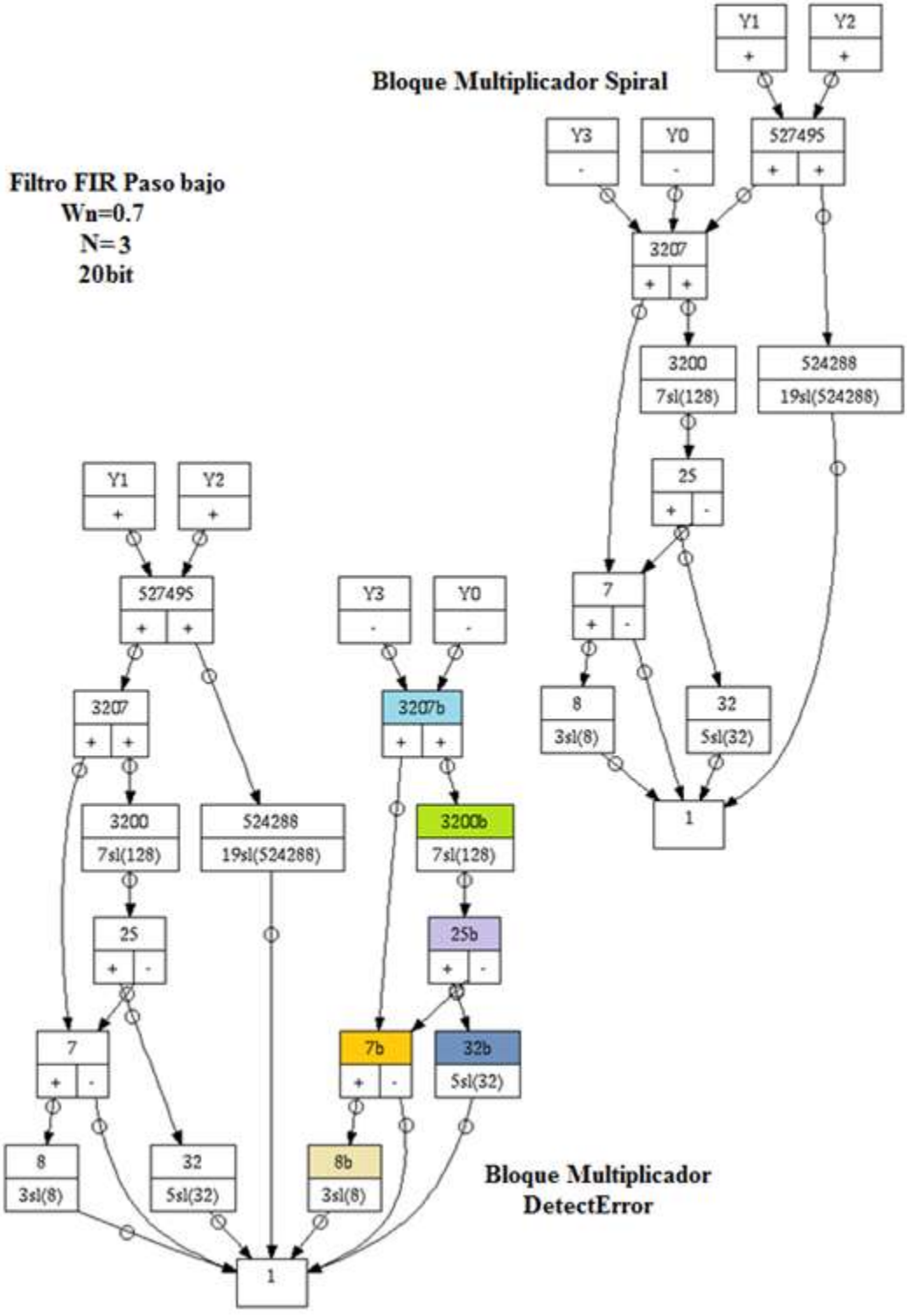


Figura 5.11 Bloques Multiplicadores Spiral y DetectError Paso bajo Orden 3.

Al realizar la Simulación digital con la herramienta ModelSim, para cada tipo de Filtro FIR Paso bajo considerado y cada entrada al mismo, se obtiene un registro con la respuesta proporcionada por ModelSim. Este registro representa el estado del Registro Check, de los nodos que alimentan las salidas, presentados en formato binario y del resto de nodos que conforman dicho Bloque, que se presentan en formato hexadecimal.

Cada uno de estos registros se debe analizar en detalle para confirmar, por comparación con el script de errores generado con la aplicación DetectError, si la indicación de Error o ausencia de error proporcionada por el Registro Check es correcta o presenta algún fallo de detección.

Para llevar a cabo a esta tarea se ha diseñado una aplicación llamada Reconcilia, diseñada e implementada en Visual Basic y cuyo objetivo es visualizar de forma muy cómoda e intuitiva la revisión de todos los errores simulados

En las Tablas 5.1 a 5.3, correspondientes a la simulación del Filtro FIR Paso bajo de Orden 4, Frecuencia normalizada de corte 0.5 y precisión de cálculo 12 bit, se recogen algunos ejemplos de estos registros, en los que se pone de manifiesto los posibles tipos de errores que se generan: Error compensado en salidas, Error confirmado y Error no detectado por desbordamiento.

En dichas tablas se resaltan en rojo las diferencias existentes en los distintos nodos después de producirse el Soft Error.

En las figuras 5.12 a 5.14 se recoge la captura de pantalla de cada uno de dichos ejemplos en la aplicación Reconcilia, confirmando visualmente como la aplicación valida el funcionamiento del Registro Check:

- Detecta Ciclos con Soft Errors y Ciclos sin error.
- No detecta Errores cuando existe desbordamiento.
- No se produce compensación en las salidas de los Bloques con nodos correctamente replicados.

Tabla 5.1 Error compensado en salidas.

X error	11010011011001101101111010110101	w257[29]
	Antes de introducir el error	**Después de introducir el error**
check	00000000000000000000000000000000000000000000	00000000000000000000000000000000000000000000
w834	11110110111010110101001000011000100110101010	11110110111001101010010000110001001101010 10
w2427	11100101100100110000010000010101110111110111	11100101100110110000010000010101110111110111
w1	fffd366deb5	fffd366deb5
w0	00000000000	00000000000
w160	fe4204b3120	fe4204b3120
w1668	edd6a431354	edceea431354
w256	fd366deb500	fd366deb500
w257	fd33a4593b5	fd31a4593b5
w4	fff4d9b7ad4	fff4d9b7ad4
w4095	d369a84714b	d369a84714b
w4096	d366deb5000	d366deb5000
w417	fb75a90c4d5	fb73a90c4d5
w5	fff21025989	fff21025989

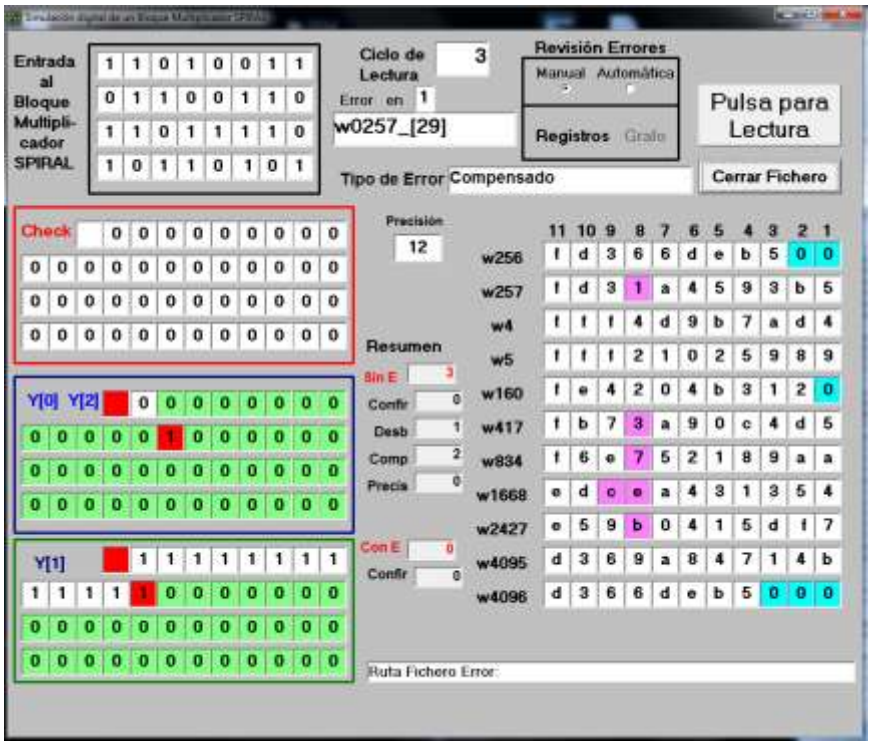


Figura 5.12 Error compensado en salidas (Reconcilia).

Tabla 5.2 Error confirmado

X error	00110001001011111011101110111111	w2427[39]
	Antes de introducir el error	**Después de introducir el error**
check	00000000000000000000000000000000000000000000	11111100000000000000000000000000000000000000
w834	00001010000000111101100000011010010000111110	00001010000000111101100000011010010000111110
w2427	00011101001001001111100010001110101111000101	00010101001001001111100010001110101111000101
w1	000312fbbbf	000312fbbbf
w0	00000000000	00000000000
w160	01ebdd55760	01ebdd55760
w1668	1407b03487c	1407b03487c
w256	0312fbbbf00	0312fbbbf00
w257	03160eb7abf	03160eb7abf
w4	000c4beeefc	000c4beeefc
w4095	312ca8c3441	312ca8c3441
w4096	312fbbbf000	312fbbbf000
w417	0501ec0d21f	0501ec0d21f
w5	000f5eeaabb	000f5eeaabb

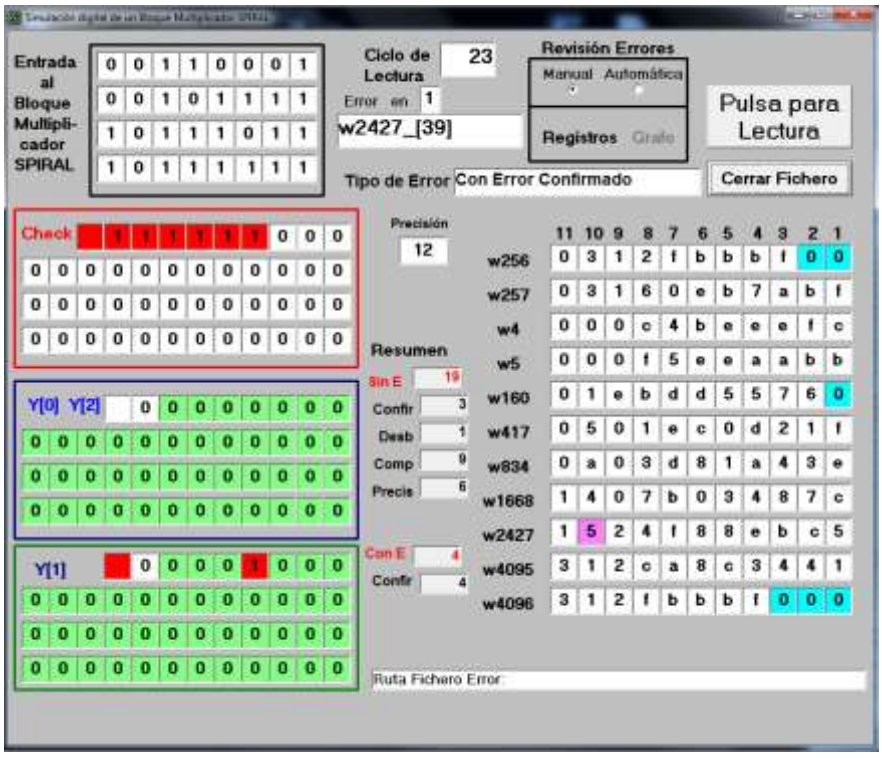


Figura 5.13 Error detectado (Reconcilia).

Tabla 5.3 Error no detectado por desbordamiento.

X error	01100010011011010010001010000001	w5[42]
	Antes de introducir el error	**Después de introducir el error**
check	00000000000000000000000000000000000000000000	00000000000000000000000000000000000000000000
w834	00010100000010100111100010100110100001000010	00010100000010100111100010100110100001000010
w2427	00111010010100100000101001100001110011111011	00111010010100100000101001100001110011111011
w1	000626d2281	000626d2281
w0	00000000000	00000000000
w160	03d843590a0	03d843590a0
w1668	2814f14d084	2814f14d084
w256	0626d228100	0626d228100
w257	062cf8fa381	062cf8fa381
w4	00189b48a04	00189b48a04
w4095	6266fbaed7f	6266fbaed7f
w4096	626d2281000	626d2281000
w417	0a053c53421	0a053c53421
w5	001ec21ac85	401ec21ac85

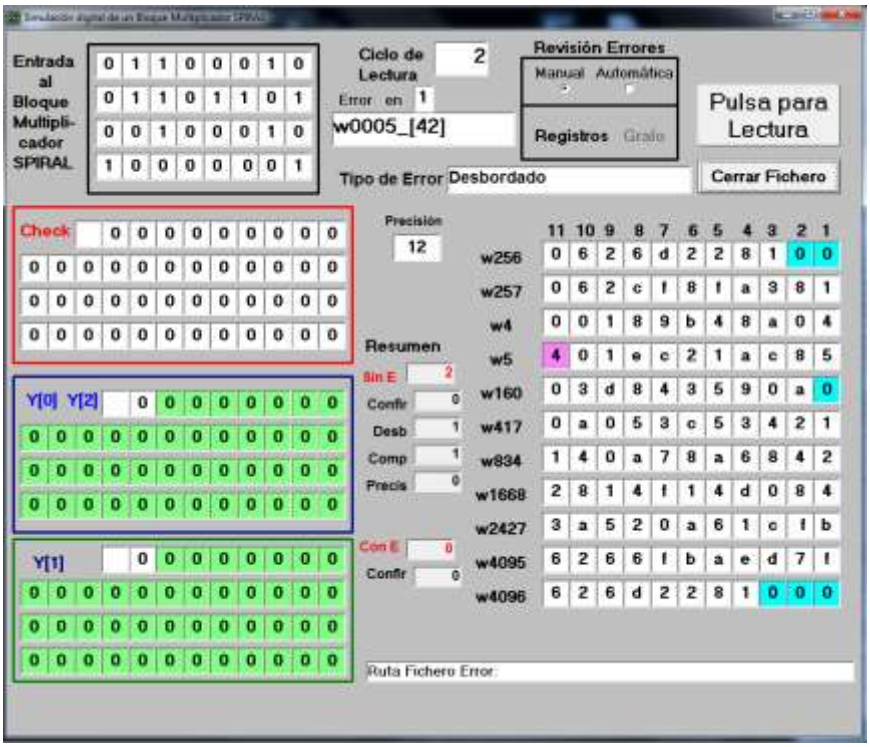


Figura 5.14 Error no detectado por desbordamiento (Reconcilia).

Resumen de la Reconciliación de Errores.

Tras inspeccionar todos los datos obtenidos en la simulación con ModelSim para los Filtros objeto de estudio, se obtienen los resúmenes indicados en las tablas 5.4, 5.5 y 5.6.

Para la simulación del Filtro FIR Paso bajo de orden 4, frecuencia normalizada 0’5 con 12 bit de precisión de cálculo, se adjunta además, en las figuras 5.15a y 5.15b la captura de pantalla correspondiente de Reconcilia en la que visualiza el último registro revisado y el resumen de errores.

Tabla 5.4 Resumen de errores Filtro FIR Paso bajo Orden 4 Frec 0’5 Precisión 12 bit

	BM Spiral	BM DetectError
Compensación de errores	SI	NO
Número total de nodos Nodos con desplazamientos Nodos a replicar	11 6 6	17 9 N.A
Características del Script		
Ciclos de entrada del Script	2789	4411
Tiempo de simulación (min)	36	43
Número de Ciclos libres de Error	589 (21,1%)	891 (20,2%)
Errores en Script	2200 (78,9%)	3520 (79,8%)
Errores que provocan desbordamiento	96 (4,4%)	175 (5,0%)
Errores que no provocan desbordamiento	2104 (95,6%)	3345 (95,0%)
Reconciliación de errores		
Número de Soft Errors detectados	923 (43,9%)	3345 (100%)
Errores no detectados por compensación	1181 (56,1%)	0 (0,0%)

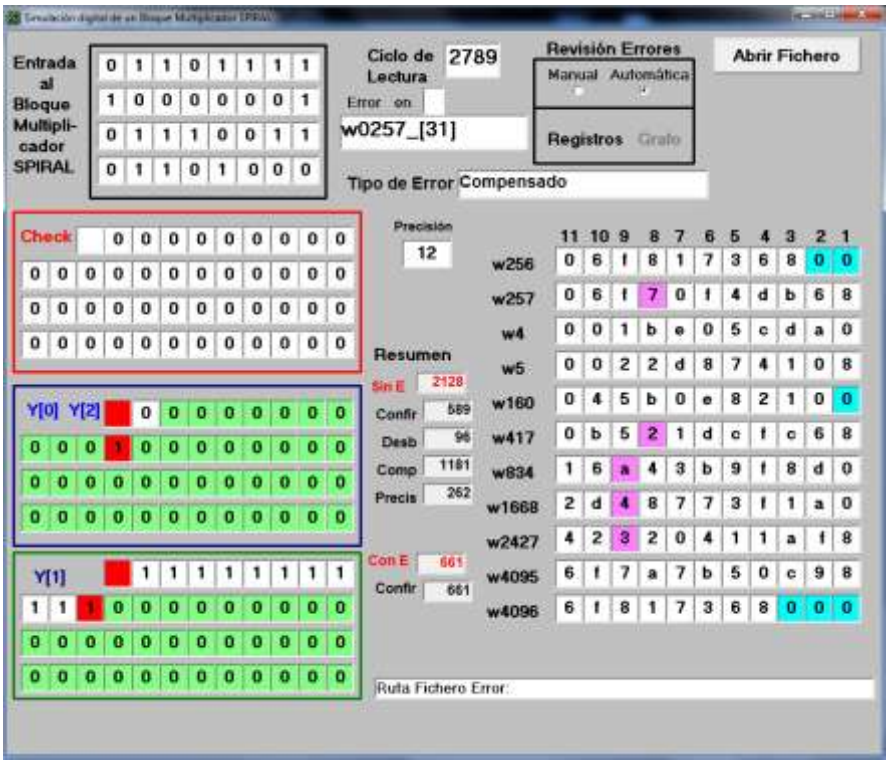


Figura 5.15a Captura Filtro FIR Paso bajo Orden 4 Frec 0'5 12bit Spiral.

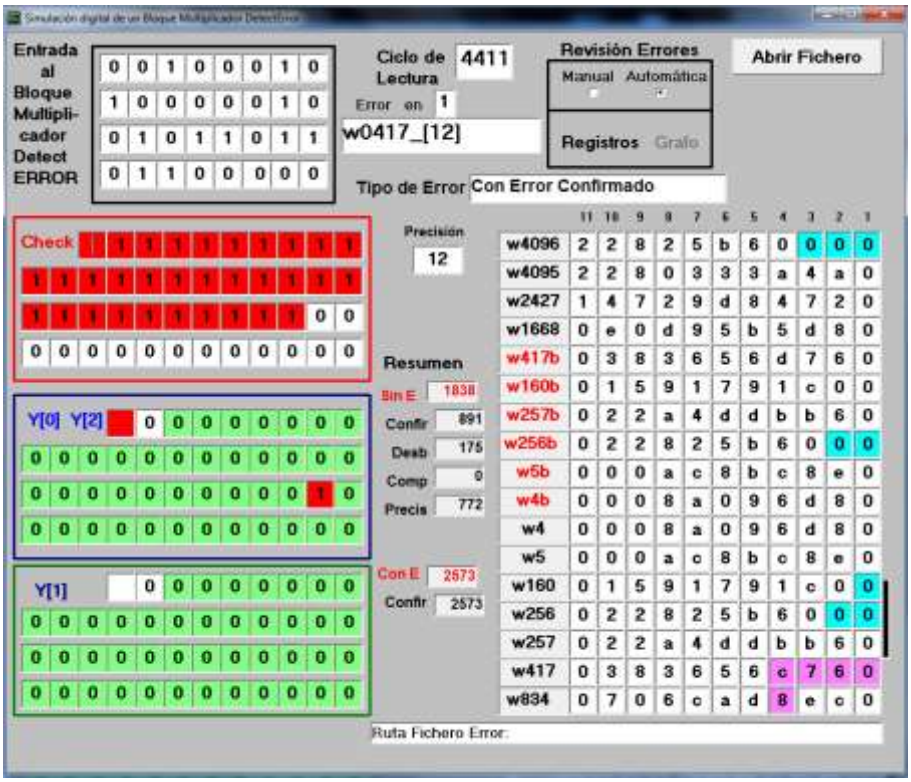


Figura 5.15b Captura Filtro FIR Paso bajo Orden 4 Frec 0'5 12bit DetectError.

Tabla 5.5 Resumen de errores Filtro FIR Paso bajo Orden 16 Frec 0'3 Precisión 8 bit

	Bloque Multiplicador Spiral	Bloque Multiplicador DetectError
Compensación de errores	SI	NO
Número total de nodos Nodos con desplazamientos Nodos a replicar	11 6 3	14 7 N.A
Características del Script		
Ciclos de entrada del Script	2517	3536
Tiempo de simulación (min)	25	35
Número de Ciclos libres de Error	517 (25,8%)	736 (20,8%)
Errores en Script	2000 (74,2%)	2800(79,2%)
Errores que provocan desbordamiento	33 (1,7%)	51 (1,8%)
Errores que no provocan desbordamiento	1967 (98,3%)	2749 (98,2%)
Reconciliación de errores		
Número de Soft Errors detectados	1218 (61,9%)	2749 (100%)
Errores no detectados por compensación	749 (38,1%)	0 (0,0%)

Tabla 5.6 Resumen de errores Filtro FIR Paso bajo Orden 3 Frec 0'7 Precisión 20 bit

	Bloque Multiplicador Spiral	Bloque Multiplicador DetectError
Compensación de errores	SI	NO
Número total de nodos Nodos con desplazamientos Nodos a replicar	8 4 6	12 7 N.A
Características del Script		
Ciclos de entrada del Script	2612	4627
Tiempo de simulación (min)	45	80
Número de Ciclos libres de Error	532 (20,4%)	987 (21,3%)
Errores en Script	2080 (79,6%)	3640(78,7%)
Errores que provocan desbordamiento	67 (3,1%)	139 (3,8%)
Errores que no provocan desbordamiento	2013 (96,9%)	3501 (96,2%)
Reconciliación de errores		
Número de Soft Errors detectados	532 (26,4%)	3501 (100%)
Errores no detectados por compensación	1481 (73,6%)	0 (0,0%)

Análisis de los resultados obtenidos.

A la vista de los datos recogidos en las tablas 5.4, 5.5 y 5.6 se obtienen las siguientes conclusiones:

Justificación de Errores.

Se confirma que todos los errores presentes en el fichero de entrada de errores generado aleatoriamente por DetectError, han sido detectados y, en función del efecto producido, son claramente asignados a cada uno de los tipos de errores definidos.

En efecto, en ambos tipos de Bloque Multiplicador se cumple que:

$$TOT = CLE + ECS + EED + ED$$

Siendo:

TOT: Número total de Ciclos de entrada.

CLE: Número de Ciclos libres de Error.

ECS: Número de Errores Compensados en Salidas.

EED: Número de Errores eliminados por desbordamiento.

ED: Número de Soft Errors detectados.

Fallos Benignos: Errores eliminados por Desbordamiento.

Este tipo de error se produce cuando el Soft Error afecta a los bits más significativos de un nodo suma/resta que alimenta un nodo de desplazamiento.

Al trasladarse el error a bits que no existen, no se produce cambio en el nodo de salida y como ésta es correcta, el Registro Check considera que no ha existido error. En realidad es un Soft Error que se autoelimina.

En las simulaciones de los Bloques Multiplicadores Spiral y DetectError de los Filtros FIR Paso bajo utilizados como ejemplo, se ha comprobado que los nodos suma/resta y bits susceptibles de originar desbordamiento, son los indicados en las tablas 5.7 y 5.8.

Tabla 5.7 Nodos con posibilidad de generar desbordamiento en Bloque Spiral.

	Bloque Multiplicador Spiral			
Nodo	w5	w5	w257	w417
Bits	43 a 39	38	43	34
Nodo desplaz	w160	w834 w1668	w834 w1668	w834 w1668

Tabla 5.8 Nodos con posibilidad de generar desbordamiento en Bloque DetectError.

	Bloque Multiplicador DetectError							
Nodo	w5	w5	w5b	w5b	w257	w257b	w417	w417b
Bits	43 a 39	38	43 a 39	38 y 37	43	43 y 42	43	43 y 42
Nodo desplaz	w160	w834	w160b	w1668	w834	w1668	w834	w1668

Teniendo en cuenta que la aparición de un Soft Error en cualquiera de dichos nodos es totalmente aleatoria, se puede estimar el porcentaje de errores que se eliminarán por desbordamiento en cada uno de los Bloques Multiplicadores, haciendo uso de la expresión:

$$\%\ \text{Errores eliminados por desbordamiento} = \frac{\text{Numero de Bits susceptibles}}{\text{Numero de Total de Bits en el Bloque}}$$

Para el Bloque Multiplicador Spiral, que cuenta con 8 bits susceptibles y 5 nodos de 44 bits, el porcentaje teórico de errores eliminados por desbordamiento ante Soft Errors totalmente aleatorios, representa un 3.63% del número total de errores producidos.

Para el Bloque Multiplicador DetectError, con una estructura algo más compleja, un total de 8 nodos de 44 bits y 19 bits susceptibles de iniciar desbordamiento, el porcentaje teórico de errores eliminados se incrementa hasta el 5.40%

En las tablas 5.9 y 5.10 se resumen los datos experimentales obtenidos al reconciliar los errores forzados y no detectados por desbordamiento en la simulación del Bloque Multiplicador Spiral y del Bloque Multiplicador DetectError. Se contabiliza 96 casos para el Bloque Spiral, lo que representa un 4.4% del total de errores y 175 casos para el Bloque DetectError (5.0% del total).

Al comparar estas cifras con los valores teóricos comentados anteriormente, se confirma que el Script de Errores utilizado para llevar a cabo la simulación digital en ModelSim se ha generado con un nivel de aleatoriedad prácticamente total.

Tabla 5.9 Reconciliación errores eliminados por desbordamiento en Bloque Spiral.

Nodo	**w5**						**w257**	**w417**
Bit	43	42	41	40	39	38	43	43
Repeticiones	7	12	14	12	15	10	12	14

Tabla 5.10 Reconciliación errores eliminados por desbordamiento en Bloque DetectError.

Nodo	**w5b**						
Bit	43	42	41	40	39	38	37
Repeticiones	12	8	8	5	12	13	10

Nodo	**w5**					
Bit	43	42	41	40	39	38
Repeticiones	6	8	6	10	14	9

Nodo	**w417b**		**w417**		**w257b**		**w257**
Bit	43	42	43		43	42	43
Repeticiones	8	10	10		8	11	7

Imposibilidad de Compensación de errores.

Se confirma que durante el desarrollo de todas las operaciones matemáticas que tienen lugar en el Bloque Multiplicador DetectError no hay posibilidad de que se produzca compensación de errores. En la simulación no se produjo ninguna compensación de errores en ninguna de las 3345 ocasiones en que se forzó la aparición de un Soft Error con ModelSim.

La frecuencia de compensación de errores en el Bloque Multiplicador estándar diseñado por Spiral supera el 56%, comprobando que en 1181 ocasiones de 2104 posibles, el error no se puede detectar por afectar de forma positiva a algunas salidas y de forma negativa a otras.

Los tres nodos con compensación de errores, localizados por DetectError son:

w5, w257 y w417.

En la tabla 5.11 se resumen los datos experimentales obtenidos al reconciliar los errores forzados y no detectados por compensación en las salidas en la simulación del BM Spiral.

Se contabilizan 367 repeticiones en el nodo w5, 427 repeticiones en el nodo w257 y 387 repeticiones en el nodo w417., los tres nodos en los que DetectError compensación de errores.

Tabla 5.11 Reconciliación errores compensados en salidas (Modelo Spiral).

Nodo	**w5**																				
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18		
Repet.	11	16	9	5	14	13	6	9	13	5	13	9	8	17	11	13	10	12	12		
Nodo	**w5**																				
Bit	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37		
Repet.	6	8	9	9	6	7	11	14	11	7	12	7	15	5	12	4	7	11	Ver nota		
Nodo	**w257**																				
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Repet.	16	8	9	8	9	14	14	4	12	11	13	10	5	10	9	9	8	12	11	14	12
Nodo	**w257**																				
Bit	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Repet.	10	14	8	4	10	13	10	9	17	8	11	12	10	11	11	10	6	9	9	15	2
Nodo	**w417**																				
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Repet.	6	8	10	9	13	9	8	11	10	7	6	15	8	12	9	6	14	11	12	10	11
Nodo	**w417**																				
Bit	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
Repet.	11	6	9	7	12	11	10	6	15	9	6	5	10	11	7	8	7	5	9	8	10

Nota: El bit 42 del nodo w417, el bit 42 del nodo w257 y el bit 37 del nodo w5 no generan errores eliminados ni por desbordamiento ni por compensación debido a las peculiares características del grafo.

En efecto, tal y como se observa en las figuras 5.16, 5.17 y 5.18, todos estos nodos presentan un comportamiento asimétrico con los dos nodos padre con los que están relacionados y que evita la eliminación de error por compensación.

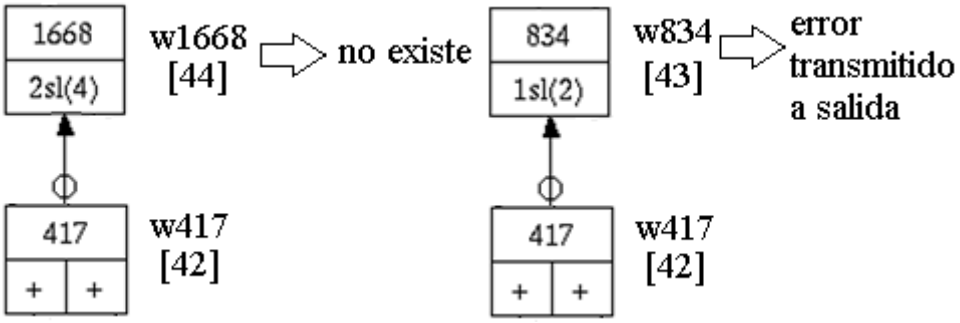


Figura 5.16 Nodo w417. Error no desbordado, no compensado.

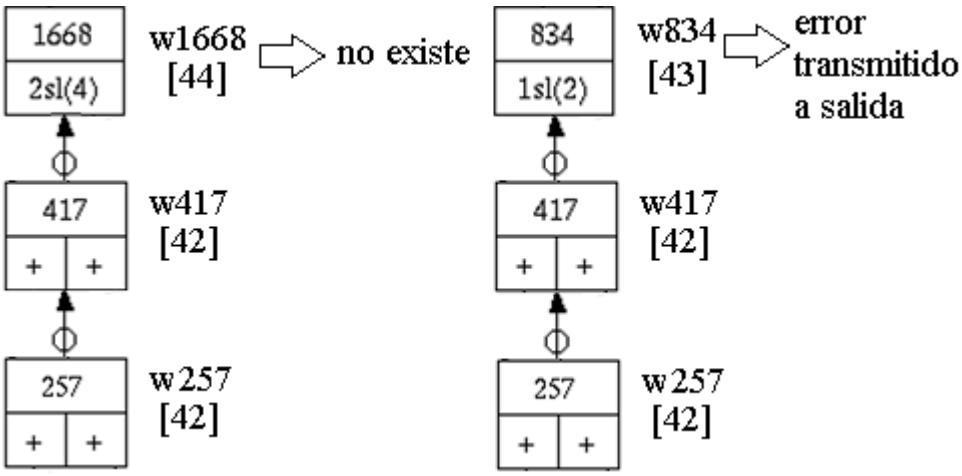


Figura 5.17 Nodo w257. Error no desbordado, no compensado.

En uno de ellos, que daría lugar a una salida con error negativo se produce desbordamiento por no existir el bit de destino del desplazamiento y en el otro nodo padre, que da lugar a una salida con error positivo, el error es detectado por el Registro Check.

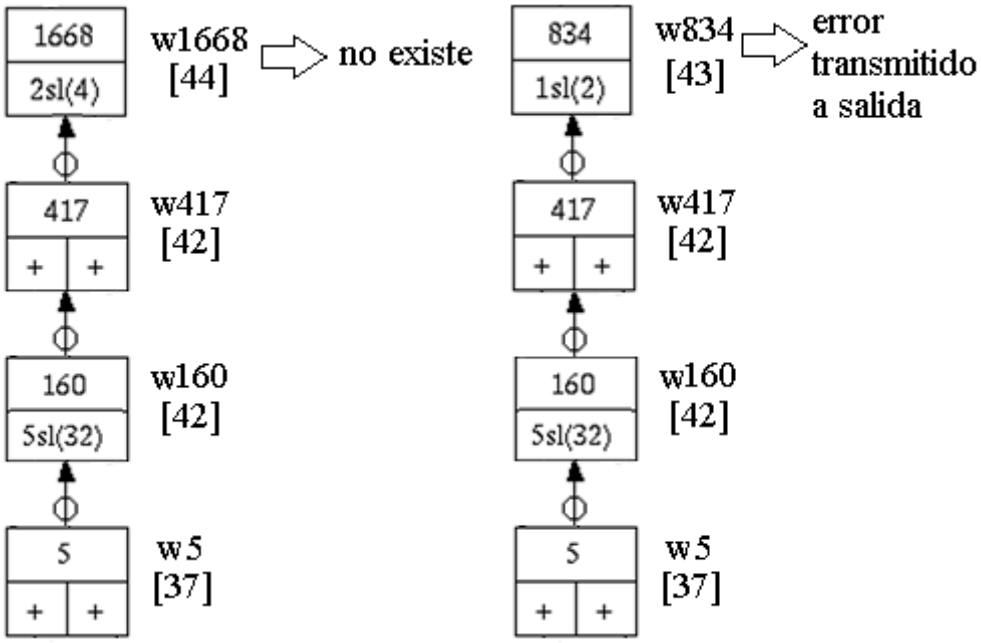


Figura 5.18 Nodo w5. Error no desbordado, no compensado.

Errores detectados según nivel de precisión exigido al Registro Check.

Para obtener el valor real de salida del Bloque Multiplicador es necesario transformar el valor entregado por cada una de las salidas dividiendo por un factor F cuyo valor es $2^{\text{fractional bit}}$, siendo fractional bit la precisión con que calculó el grafo con el programa Spiral.

Si se modifica este valor F y se hace mayor, se indica al Check que identifique como lectura Sin Error, en realidad Error Eliminado por Precisión cualquier Error cuya suma presente diferencia en un bit menos significativo del Registro Check que el fractionalbit de diseño del filtro.

Por el contrario si se modifica dicho valor haciéndolo más pequeño, se impone al Registro Check que identifique como lectura Con Error cualquier Soft Error cuya suma presente diferencia en un bit más significativo del Registro Check que el fractionalbit de diseño del filtro.

En la tabla 5.12, donde se presenta la reconciliación de errores de la simulación realizada con la herramienta ModelSim para distintos valores de bit de precisión, se confirma dicho razonamiento.

Mientras el número de ciclos sin error, el número de errores eliminados por compensación y el número de errores eliminados por desbordamiento es siempre el mismo, a medida que aumenta el

bit de precisión utilizado para llevar a cabo la revisión de errores, es más pequeño el número de Errores que confirma como tales el Registro Check.

En la gráfica 5.19, se representa como varía para cada valor de bit de precisión, el número de Errores Confirmados y Errores Eliminados por Precisión que identifica el Registro Check.

Tabla 5.12 Reconciliación errores según bit de precisión para Bloque Spiral.

Bit de precisión	1	2	3	4	5	6	7	8	9	10	11
Sin Error	1895	1915	1945	1973	1984	2002	2018	2042	2063	2084	2108
Confirmado	589	589	589	589	589	589	589	589	589	589	589
Desbordado	96	96	96	96	96	96	96	96	96	96	96
Compensado	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181
Por Precisión	29	49	79	107	118	136	152	176	197	218	242
Con Error	894	874	844	816	805	787	771	747	726	705	681
Bit de precisión	12	13	14	15	16	17	18	19	20	21	22
Sin Error	2128	2146	2169	2194	2204	2231	2250	2267	2285	2306	2319
Confirmado	589	589	589	589	589	589	589	589	589	589	589
Desbordado	96	96	96	96	96	96	96	96	96	96	96
Compensado	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181
Por Precisión	262	280	303	328	338	365	384	401	419	440	448
Con error	661	643	620	595	585	558	539	522	504	483	470
Bit de precisión	23	24	25	26	27	28	29	30	31	32	33
Sin Error	2343	2361	2379	2398	2416	2440	2462	2480	2496	2519	2543
Confirmado	589	589	589	589	589	589	589	589	589	589	589
Desbordado	96	96	96	96	96	96	96	96	96	96	96
Compensado	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181
Por Precisión	465	484	501	525	537	562	590	603	623	642	677
Con error	446	428	410	391	373	349	327	309	293	270	246
Bit de precisión	34	35	36	37	38	39	40	41	42	43	44
Sin Error	2561	2580	2596	2614	2632	2653	2683	2703	2728	2754	2789
Confirmado	589	589	589	589	589	589	589	589	589	589	589
Desbordado	96	96	96	96	96	96	96	96	96	96	96
Compensado	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181	1181
Por Precisión	695	714	730	748	766	787	817	837	862	888	923
Con Error	228	209	193	175	157	136	106	86	61	35	0

Figura 5.19 Errores Confirmados y Errores Eliminados por Precisión.

Capítulo 6 - Resumen y conclusiones.

6.1 Resumen de resultados.

En este capítulo se recogen los resultados obtenidos para cada uno de las tareas del proyecto, indicadas en el capítulo 2 de esta Memoria.

- ***Propuesta e implantación del Método de Detección de Errores en Bloques Multiplicadores de Filtro FIR sin posibilidad de experimentar Soft Errors indetectables.***

Tras confirmar la imposibilidad de detectar la aparición de Soft Errors en los Bloques Multiplicadores estándar de Filtros FIR generados online mediante el uso del programa Spiral, se ha materializado un nuevo esquema de diseño de dichos bloques que pone fácilmente de manifiesto la aparición de cualquier Soft Error durante la realización de las operaciones aritméticas generadas en su interior.

El nuevo esquema propuesto, bautizado como Método DetectError, imposibilita la compensación de errores en el interior del Bloque Multiplicador, al diseñar de forma óptima su estructura asegurando inequívocamente que cualquier error generado es detectado.

Una vez establecido el fundamento matemático en que se apoya el Método DetectError, se ha completado con éxito el diseño del Software necesario para implementar de forma automática nodos sin compensación de errores y generar por tanto grafos con dicha funcionalidad para cualquier Bloque Multiplicador.

- ***Evaluación de Área del Método de Detección propuesto y comparativa con la técnica DMR.***

Tras determinar los tamaños del circuito ASIC necesario para hacer operativo un Filtro FIR sin compensación de errores DetectError, así como los correspondientes valores de tamaños de circuitos de las otras opciones existentes de detección de Soft Errors (Bloque Spiral estándar y Redundancia Dual Modular), se confirma que el método propuesto en este trabajo de Máster en Investigación en Informática, el Método DetectError requiere tamaños de circuito hasta un 30% más pequeños, presentando con ello gran ventaja económica y de prestaciones sobre sus competidores.

Salvo para un 18% de situaciones de Bloques Multiplicadores de Filtros FIR Paso bajo de 8 bit de y un 3% de situaciones de Bloques Multiplicadores de Filtros FIR Paso alto de 8 bit, el

Método DetectError ofrece siempre Áreas menores que las proporcionadas por los métodos utilizados actualmente para detectar la aparición de Soft Errors.

- ***Comprobación mediante ModelSim de la eficacia de la técnica de Replicación Parcial en la detección de Soft Errors.***

Respetando el protocolo establecido por la IEEE, se ha confirmado mediante simulación digital, utilizando para ello la herramienta ModelSim, que el Método DetectError asegura totalmente la detección de Soft Errors habida cuenta de la nueva estructura propuesta para el diseño de Bloques Multiplicadores de Filtros FIR.

Además este nuevo Método permite establecer el nivel de aceptación o rechazo de un Soft Error detectado, al poder establecerse fácilmente el nivel de precisión de corte de dicho evento. De esta forma se evita el coste asociado a la repetición de cálculos, que de otra forma sería necesario repetir al ser catalogados dichos eventos como inductores de error, pero que en la práctica pueden ser ignorados, dado que su contribución al valor final de salida del Filtro no sobrepasa la precisión de cálculo del Bloque Multiplicador.

6.2 Conclusión Final de este Proyecto.

A la vista de los resultados descritos en el punto 6.1, se confirma la consecución del objetivo propuesto en el capítulo 2 de este Trabajo de Fin de Master en Ingeniería Informática.

El Método DetectError es la estrategia más sencilla y efectiva de diseño para asegurar la detección inequívoca de Soft Errors en los Bloques Multiplicadores de Filtros FIR Paso bajo y Paso alto.

ANEXOS

Anexo A- 1: Coeficientes utilizados en Filtros FIR

A-1.1 Coeficientes MATLAB para Filtros FIR Paso bajo.

Orden (N): 2 a 16 y Frecuencia normalizada de corte (Wn): 0.1 a 0.9

	Wn	0.10			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0680	0,0457	0,0338	0,0264	0,0212	0,0174	0,0144	0,0120	0,0100	0,0083	0,0068	0,0055	0,0044	0,0034	0,0025
B(1)	0,8640	0,4543	0,2401	0,1405	0,0897	0,0612	0,0439	0,0326	0,0249	0,0193	0,0152	0,0120	0,0095	0,0074	0,0057
B(2)	0,0680	0,4543	0,4521	0,3331	0,2343	0,1662	0,1202	0,0888	0,0668	0,0511	0,0395	0,0308	0,0241	0,0188	0,0147
B(3)		0,0457	0,2401	0,3331	0,3094	0,2552	0,2025	0,1590	0,1249	0,0985	0,0780	0,0620	0,0495	0,0395	0,0315
B(4)			0,0338	0,1405	0,2343	0,2552	0,2380	0,2076	0,1756	0,1464	0,1212	0,1000	0,0823	0,0677	0,0555
B(5)				0,0264	0,0897	0,1662	0,2025	0,2076	0,1957	0,1765	0,1553	0,1346	0,1155	0,0984	0,0834
B(6)					0,0212	0,0612	0,1202	0,1590	0,1756	0,1765	0,1682	0,1552	0,1402	0,1248	0,1099
B(7)						0,0174	0,0439	0,0888	0,1249	0,1464	0,1553	0,1552	0,1493	0,1400	0,1289
B(8)							0,0144	0,0326	0,0668	0,0985	0,1212	0,1346	0,1402	0,1400	0,1358
B(9)								0,0120	0,0249	0,0511	0,0780	0,1000	0,1155	0,1248	0,1289
B(10)									0,0100	0,0193	0,0395	0,0620	0,0823	0,0984	0,1099
B(11)										0,0083	0,0152	0,0308	0,0495	0,0677	0,0834
B(12)											0,0068	0,0120	0,0241	0,0395	0,0555
B(13)												0,0055	0,0095	0,0188	0,0315
B(14)													0,0044	0,0074	0,0147
B(15)														0,0034	0,0057
B(16)															0,0025

	Wn	0.20			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0651	0,0416	0,0284	0,0197	0,0135	0,0088	0,0051	0,0022	0,0000	-0,002	-0,003	-0,003	-0,004	-0,003	-0,003
B(1)	0,8698	0,4584	0,2370	0,1324	0,0785	0,0479	0,0294	0,0174	0,0093	0,0038	0,0000	-0,003	-0,004	-0,005	-0,005
B(2)	0,0651	0,4584	0,4692	0,3479	0,2409	0,1640	0,1107	0,0737	0,0476	0,0290	0,0158	0,0065	0,000	-0,004	-0,007
B(3)		0,0416	0,2370	0,3479	0,3344	0,2793	0,2193	0,1662	0,1224	0,0872	0,0594	0,0378	0,0213	0,0089	0,0000
B(4)			0,0284	0,1324	0,2409	0,2793	0,2710	0,2405	0,2022	0,1633	0,1271	0,0949	0,0673	0,0442	0,0255
B(5)				0,0197	0,0785	0,1640	0,2193	0,2405	0,2370	0,2183	0,1914	0,1609	0,1299	0,1002	0,0731
B(6)					0,0135	0,0479	0,1107	0,1662	0,2022	0,2183	0,2180	0,2058	0,1854	0,1601	0,1325
B(7)						0,0088	0,0294	0,0737	0,1224	0,1633	0,1914	0,2058	0,2076	0,1991	0,1826
B(8)							0,0051	0,0174	0,0476	0,0872	0,1271	0,1609	0,1854	0,1991	0,2023
B(9)								0,0022	0,0093	0,0290	0,0594	0,0949	0,1299	0,1601	0,1826
B(10)									0,0000	0,0038	0,0158	0,0378	0,0673	0,1002	0,1325
B(11)										-0,002	0,0000	0,0065	0,0213	0,0442	0,0731
B(12)											-0,003	-0,003	0,0000	0,0089	0,0255
B(13)												-0,003	-0,004	-0,004	0,0000
B(14)													-0,004	-0,005	-0,007
B(15)														-0,003	-0,005
B(16)															-0,003

	Wn	0.30			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0604	0,035	0,0201	0,0102	0,0033	-0,001	-0,004	-0,005	-0,005	-0,004	-0,002	-6E-04	0,0011	0,0024	0,003
B(1)	0,8792	0,465	0,2309	0,1177	0,059	0,0264	0,0077	-0,003	-0,008	-0,01	-0,009	-0,007	-0,004	-9E-04	0,0016
B(2)	0,0604	0,465	0,4981	0,3721	0,2492	0,1558	0,0893	0,0435	0,0134	-0,005	-0,015	-0,017	-0,016	-0,012	-0,007
B(3)		0,035	0,2309	0,3721	0,377	0,3192	0,2433	0,1695	0,1057	0,055	0,0177	-0,007	-0,02	-0,025	-0,023
B(4)			0,0201	0,1177	0,2492	0,3192	0,3276	0,2951	0,2405	0,1778	0,1166	0,0631	0,021	-0,008	-0,025
B(5)				0,0102	0,059	0,1558	0,2433	0,2951	0,3072	0,2861	0,2417	0,1846	0,1245	0,069	0,0234
B(6)					0,0033	0,0264	0,0893	0,1695	0,2405	0,2861	0,3001	0,2841	0,2445	0,1902	0,1303
B(7)						-0,001	0,0077	0,0435	0,1057	0,1778	0,2417	0,2841	0,2984	0,2846	0,2473
B(8)							-0,004	-0,003	0,0134	0,055	0,1166	0,1846	0,2445	0,2846	0,2985
B(9)								-0,005	-0,008	-0,005	0,0177	0,0631	0,1245	0,1902	0,2473
B(10)									-0,005	-0,01	-0,015	-0,007	0,021	0,069	0,1303
B(11)										-0,004	-0,009	-0,017	-0,02	-0,008	0,0234
B(12)											-0,002	-0,007	-0,016	-0,025	-0,025
B(13)												-6E-04	-0,004	-0,012	-0,023
B(14)													0,0011	-9E-04	-0,007
B(15)														0,0024	0,0016
B(16)															0,003

	Wn	0.40			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,054	0,0265	0,0101	0	-0,005	-0,007	-0,006	-0,003	0	0,0027	0,004	0,0037	0,0021	0	-0,002
B(1)	0,892	0,4735	0,2203	0,0952	0,0317	0	-0,014	-0,016	-0,013	-0,006	0	0,0045	0,0063	0,0056	0,0031
B(2)	0,054	0,4735	0,5391	0,4048	0,255	0,1351	0,0512	0	-0,025	-0,03	-0,023	-0,012	0	0,0079	0,0108
B(3)		0,0265	0,2203	0,4048	0,4375	0,3721	0,2657	0,1555	0,0635	0	-0,033	-0,042	-0,033	-0,017	0
B(4)			0,0101	0,0952	0,255	0,3721	0,4057	0,3641	0,2748	0,1687	0,0716	0	-0,04	-0,051	-0,041
B(5)				0	0,0317	0,1351	0,2657	0,3641	0,3981	0,3649	0,2824	0,1776	0,0771	0	-0,045
B(6)					-0,005	0	0,0512	0,1555	0,2748	0,3649	0,3976	0,3674	0,288	0,1839	0,081
B(7)						-0,007	-0,014	0	0,0635	0,1687	0,2824	0,3674	0,3987	0,37	0,2924
B(8)							-0,006	-0,016	-0,025	0	0,0716	0,1776	0,288	0,37	0,4003
B(9)								-0,003	-0,013	-0,03	-0,033	0	0,0771	0,1839	0,2924
B(10)									0	-0,006	-0,023	-0,042	-0,04	0	0,081
B(11)										0,0027	0	-0,012	-0,033	-0,051	-0,045
B(12)											0,004	0,0045	0	-0,017	-0,041
B(13)												0,0037	0,0063	0,0079	0
B(14)													0,0021	0,0056	0,0108
B(15)														0	0,0031
B(16)															-0,002

	Wn	0.50			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0462	0,0167	0	-0,008	-0,009	-0,005	0	0,004	0,0051	0,0033	0	-0,003	-0,004	-0,002	0
B(1)	0,9076	0,4833	0,2037	0,0645	0	-0,023	-0,023	-0,012	0	0,0076	0,009	0,0054	0	-0,004	-0,005
B(2)	0,0462	0,4833	0,5926	0,4433	0,2518	0,0968	0	-0,041	-0,042	-0,022	0	0,014	0,0162	0,0095	0
B(3)		0,0167	0,2037	0,4433	0,5138	0,4313	0,274	0,1147	0	-0,054	-0,057	-0,031	0	0,02	0,0232
B(4)			0	0,0645	0,2518	0,4313	0,4974	0,4344	0,2885	0,1257	0	-0,063	-0,068	-0,038	0
B(5)				-0,008	0	0,0968	0,274	0,4344	0,4968	0,44	0,2984	0,133	0	-0,07	-0,076
B(6)					-0,009	-0,023	0	0,1147	0,2885	0,44	0,4996	0,4451	0,3049	0,1374	0
B(7)						-0,005	-0,023	-0,041	0	0,1257	0,2984	0,4451	0,5019	0,4472	0,3077
B(8)							0	-0,012	-0,042	-0,054	0	0,133	0,3049	0,4472	0,5009
B(9)								0,004	0	-0,022	-0,057	-0,063	0	0,1374	0,3077
B(10)									0,0051	0,0076	0	-0,031	-0,068	-0,07	0
B(11)										0,0033	0,009	0,014	0	-0,038	-0,076
B(12)											0	0,0054	0,0162	0,02	0
B(13)												-0,003	0	0,0095	0,0232
B(14)													-0,004	-0,004	0
B(15)														-0,002	-0,005
B(16)															-0.0000

	Wn	0.60			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0373	0,0065	-0,008	-0,01	-0,005	0,0022	0,006	0,0046	0	-0,004	-0,004	-0,001	0,0021	0,0034	0,0019
B(1)	0,9253	0,4935	0,1793	0,0269	-0,029	-0,032	-0,013	0,0052	0,0127	0,0088	0	-0,006	-0,006	-0,002	0,0031
B(2)	0,0373	0,4935	0,6579	0,4836	0,2335	0,0418	-0,05	-0,058	-0,025	0,0098	0,0236	0,016	0	-0,011	-0,011
B(3)		0,0065	0,1793	0,4836	0,6011	0,488	0,2598	0,0502	-0,064	-0,077	-0,034	0,0137	0,0331	0,0227	0
B(4)			-0,008	0,0269	0,2335	0,488	0,5951	0,4982	0,276	0,0553	-0,072	-0,09	-0,04	0,0165	0,0407
B(5)				-0,01	-0,029	0,0418	0,2598	0,4982	0,5997	0,5071	0,2853	0,0581	-0,077	-0,098	-0,045
B(6)					-0,005	-0,032	-0,05	0,0502	0,276	0,5071	0,6027	0,5094	0,2887	0,0596	-0,081
B(7)						0,0022	-0,013	-0,058	-0,064	0,0553	0,2853	0,5094	0,5995	0,5082	0,2913
B(8)							0,006	0,0052	-0,025	-0,077	-0,072	0,0581	0,2887	0,5082	0,5982
B(9)								0,0046	0,0127	0,0098	-0,034	-0,09	-0,077	0,0596	0,2913
B(10)									0	0,0088	0,0236	0,0137	-0,04	-0,098	-0,081
B(11)										-0,004	0	0,016	0,0331	0,0165	-0,045
B(12)											-0,004	-0,006	0	0,0227	0,0407
B(13)												-0,001	-0,006	-0,011	0
B(14)													0,0021	-0,002	-0,011
B(15)														0,0034	0,0031
B(16)															0,0019

	Wn	0.70			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0278	-0,003	-0,013	-0,007	0,0026	0,0071	0,0037	-0,003	-0,005	-0,002	0,0025	0,0039	0,0011	-0,002	-0,003
B(1)	0,9444	0,5031	0,1458	-0,013	-0,047	-0,023	0,007	0,0169	0,0079	-0,005	-0,009	-0,003	0,0039	0,0058	0,0016
B(2)	0,0278	0,5031	0,7338	0,5205	0,1967	-0,021	-0,081	-0,042	0,0131	0,0314	0,0145	-0,009	-0,016	-0,006	0,0067
B(3)		-0,003	0,1458	0,5205	0,6945	0,5366	0,222	-0,026	-0,104	-0,055	0,0177	0,0434	0,0204	-0,013	-0,023
B(4)			-0,013	-0,013	0,1967	0,5366	0,6974	0,5528	0,2361	-0,028	-0,116	-0,063	0,021	0,0529	0,0253
B(5)				-0,007	-0,047	-0,021	0,222	0,5528	0,7036	0,5583	0,2413	-0,029	-0,125	-0,069	0,0236
B(6)					0,0026	-0,023	-0,081	-0,026	0,2361	0,5583	0,6989	0,5575	0,2453	-0,03	-0,131
B(7)						0,0071	0,007	-0,042	-0,104	-0,028	0,2413	0,5575	0,6985	0,5624	0,2493
B(8)							0,0037	0,0169	0,0131	-0,055	-0,116	-0,029	0,2453	0,5624	0,7023
B(9)								-0,003	0,0079	0,0314	0,0177	-0,063	-0,125	-0,03	0,2493
B(10)									-0,005	-0,005	0,0145	0,0434	0,021	-0,069	-0,131
B(11)										-0,002	-0,009	-0,009	0,0204	0,0529	0,0236
B(12)											0,0025	-0,003	-0,016	-0,013	0,0253
B(13)												0,0039	0,0039	-0,006	-0,023
B(14)													0,0011	0,0058	0,0067
B(15)														-0,002	0,0016
B(16)															-0,003

	Wn	0.80			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,018	-0,011	-0,012	0	0,008	0,0043	-0,004	-0,005	0	0,0044	0,0025	-0,002	-0,003	0	0,003
B(1)	0,9639	0,5109	0,1033	-0,049	-0,046	0	0,0218	0,0101	-0,008	-0,01	0	0,0073	0,0039	-0,003	-0,005
B(2)	0,018	0,5109	0,8181	0,5494	0,1426	-0,08	-0,082	0	0,0402	0,0186	-0,014	-0,019	0	0,0128	0,0067
B(3)		-0,011	0,1033	0,5494	0,7917	0,5755	0,1625	-0,097	-0,103	0	0,0543	0,0259	-0,021	-0,027	0
B(4)			-0,012	-0,049	0,1426	0,5755	0,8031	0,5919	0,1708	-0,105	-0,116	0	0,0651	0,0315	-0,025
B(5)				0	-0,046	-0,08	0,1625	0,5919	0,8005	0,5918	0,175	-0,11	-0,126	0	0,0721
B(6)					0,008	0	-0,082	-0,097	0,1708	0,5918	0,7976	0,5983	0,1792	-0,114	-0,131
B(7)						0,0043	0,0218	0	-0,103	-0,105	0,175	0,5983	0,8028	0,5998	0,1801
B(8)							-0,004	0,0101	0,0402	0	-0,116	-0,11	0,1792	0,5998	0,7979
B(9)								-0,005	-0,008	0,0186	0,0543	0	-0,126	-0,114	0,1801
B(10)									0	-0,01	-0,014	0,0259	0,0651	0	-0,131
B(11)										0,0044	0	-0,019	-0,021	0,0315	0,0721
B(12)											0,0025	0,0073	0	-0,027	-0,025
B(13)												-0,002	0,0039	0,0128	0
B(14)													-0,003	-0,003	0,0067
B(15)														0	-0,005
B(16)															0,003

	Wn	0.90			COEFICIENTES DEL FILTRO FIR PASO-BAJO DE ORDEN N										
B(i)/N	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
B(0)	0,0086	-0,016	-0,008	0,0071	0,0068	-0,003	-0,006	0,0009	0,0051	0,0007	-0,004	-0,002	0,0029	0,0024	-0,002
B(1)	0,9828	0,5161	0,0536	-0,074	-0,029	0,0229	0,0185	-0,008	-0,013	0,0017	0,0091	0,0012	-0,006	-0,003	0,0042
B(2)	0,0086	0,5161	0,9079	0,5673	0,0752	-0,122	-0,051	0,0414	0,034	-0,014	-0,024	0,0031	0,0161	0,0021	-0,011
B(3)		-0,016	0,0536	0,5673	0,8936	0,6023	0,0856	-0,145	-0,064	0,0545	0,0465	-0,02	-0,033	0,0044	0,0232
B(4)			-0,008	-0,074	0,0752	0,6023	0,9055	0,611	0,0893	-0,159	-0,072	0,0634	0,055	-0,024	-0,041
B(5)				0,0071	-0,029	-0,122	0,0856	0,611	0,8956	0,6164	0,0926	-0,168	-0,077	0,0691	0,0616
B(6)					0,0068	0,0229	-0,051	-0,145	0,0893	0,6164	0,9034	0,6217	0,0936	-0,172	-0,081
B(7)						-0,003	0,0185	0,0414	-0,064	-0,159	0,0926	0,6217	0,8977	0,6209	0,0951
B(8)							-0,006	-0,008	0,034	0,0545	-0,072	-0,168	0,0936	0,6209	0,9013
B(9)								0,0009	-0,013	-0,014	0,0465	0,0634	-0,077	-0,172	0,0951
B(10)									0,0051	0,0017	-0,024	-0,02	0,055	0,0691	-0,081
B(11)										0,0007	0,0091	0,0031	-0,033	-0,024	0,0616
B(12)											-0,004	0,0012	0,0161	0,0044	-0,041
B(13)												-0,002	-0,006	0,0021	0,0232
B(14)													0,0029	-0,003	-0,011
B(15)														0,0024	0,0042
B(16)															-0,002

A-1.2 Coeficientes finalesos para Filtros FIR Paso alto.

	Wn	0,1	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,00445	-0,00480	-0,00507	-0,00522	-0,00521	-0,00249	-0,00223	-0,00178
B(1)	0,50890	-0,03410	-0,02142	-0,01590	-0,01299	-0,00556	-0,00480	-0,00402
B(2)	-0,00445	0,57781	-0,05596	-0,04357	-0,03492	-0,01446	-0,01220	-0,01028
B(3)		-0,03410	0,66491	-0,07340	-0,06525	-0,02857	-0,02508	-0,02199
B(4)		-0,00480	-0,05596	0,77617	-0,09173	-0,04440	-0,04174	-0,03879
B(5)			-0,02142	-0,07340	0,92018	-0,05689	-0,05856	-0,05834
B(6)			-0,00507	-0,04357	-0,09173	0,55474	-0,07107	-0,07683
B(7)				-0,01590	-0,06525	-0,05689	0,68135	-0,09009
B(8)				-0,00522	-0,03492	-0,04440	-0,07107	0,85424
B(9)					-0,01299	-0,02857	-0,05856	-0,09009
B(10)					-0,00521	-0,01446	-0,04174	-0,07683
B(11)						-0,00556	-0,02508	-0,05834
B(12)						-0,00249	-0,01220	-0,03879
B(13)							-0,00480	-0,02199
B(14)							-0,00223	-0,01028
B(15)								-0,00402
B(16)								-0,00178

	Wn	0,2	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,00972	-0,01055	-0,01004	-0,00357	0,00000	0,00188	0,00294	0,00209
B(1)	0,51944	-0,08805	-0,05838	-0,02068	-0,00628	0,00000	0,00333	0,00343
B(2)	-0,00972	0,69721	-0,17924	-0,07802	-0,03213	-0,01095	0,00000	0,00462
B(3)		-0,08805	0,99533	-0,15452	-0,08264	-0,04115	-0,01742	0,00000
B(4)		-0,01055	-0,17924	0,76358	-0,13660	-0,08801	-0,05517	-0,01741
B(5)			-0,05838	-0,15452	0,64031	-0,13257	-0,10651	-0,04982
B(6)			-0,01004	-0,07802	-0,13660	0,60409	-0,15198	-0,09030
B(7)				-0,02068	-0,08264	-0,13257	0,68085	-0,12448
B(8)				-0,00357	-0,03213	-0,08801	-0,15198	0,55157
B(9)					-0,00628	-0,04115	-0,10651	-0,12448
B(10)					0,00000	-0,01095	-0,05517	-0,09030
B(11)						0,00000	-0,01742	-0,04982
B(12)						0,00188	0,00000	-0,01741
B(13)							0,00333	0,00000
B(14)							0,00294	0,00462
B(15)								0,00343
B(16)								0,00209

	Wn	0,3	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,01564	-0,01523	-0,00321	0,00277	0,00677	0,00285	0,00335	0,00958
B(1)	0,53127	-0,17485	-0,05741	-0,00522	0,01044	0,01032	-0,01168	0,00512
B(2)	-0,01564	0,88014	-0,24261	-0,06061	-0,01734	0,01659	-0,04810	-0,02119
B(3)		-0,17485	0,85646	-0,16523	-0,13729	-0,02026	-0,06109	-0,07333
B(4)		-0,01523	-0,24261	0,51908	-0,31232	-0,13337	0,06285	-0,07994
B(5)			-0,05741	-0,16523	0,93074	-0,27652	0,37347	0,07430
B(6)			-0,00321	-0,06061	-0,31232	0,80103	0,73348	0,41452
B(7)				-0,00522	-0,13729	-0,27652	-2,08894	0,78649
B(8)				0,00277	-0,01734	-0,13337	0,73348	-2,21545
B(9)					0,01044	-0,02026	0,37347	0,78649
B(10)					0,00677	0,01659	0,06285	0,41452
B(11)						0,01032	-0,06109	0,07430
B(12)						0,00285	-0,04810	-0,07994
B(13)							-0,01168	-0,07333
B(14)							0,00335	-0,02119
B(15)								0,00512
B(16)								0,00958

	Wn	0,4	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,02195	-0,00725	0,00727	0,00677	0,00000	0,01050	0,00530	0,00222
B(1)	0,54391	-0,15835	-0,04226	0,01498	-0,04078	0,00000	0,01569	-0,00364
B(2)	-0,02195	0,58121	-0,33971	-0,05649	-0,07966	-0,06102	0,00000	-0,01284
B(3)		-0,15835	0,87440	-0,29291	0,20488	-0,08759	-0,08206	0,00000
B(4)		-0,00725	-0,33971	0,67093	0,88654	0,18735	-0,09925	0,04845
B(5)			-0,04226	-0,29291	-1,92631	0,73885	0,19162	0,05294
B(6)			0,00727	-0,05649	0,88654	-1,56055	0,71584	-0,09595
B(7)				0,01498	0,20488	0,73885	-1,48648	-0,34630
B(8)				0,00677	-0,07966	0,18735	0,71584	0,71125
B(9)					-0,04078	-0,08759	0,19162	-0,34630
B(10)					0,00000	-0,06102	-0,09925	-0,09595
B(11)						0,00000	-0,08206	0,05294
B(12)						0,01050	0,00000	0,04845
B(13)							0,01569	0,00000
B(14)							0,00530	-0,01284
B(15)								-0,00364
B(16)								0,00222

	Wn	0,5	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,02835	0,00000	0,00991	0,00000	0,01234	0,00000	0,00381	0,00000
B(1)	0,55671	-0,27506	0,00000	-0,06742	0,00000	0,02129	0,00000	0,00588
B(2)	-0,02835	0,80013	-0,28602	0,00000	-0,10225	0,00000	-0,01687	0,00000
B(3)		-0,27506	0,58347	0,81504	0,00000	-0,13530	0,00000	-0,02605
B(4)		0,00000	-0,28602	-1,47961	0,70331	0,00000	0,07134	0,00000
B(5)			0,00000	0,81504	-1,21117	0,70535	0,00000	0,08540
B(6)			0,00991	0,00000	0,70331	-1,18073	-0,31801	0,00000
B(7)				-0,06742	0,00000	0,70535	0,52336	-0,34527
B(8)				0,00000	-0,10225	0,00000	-0,31801	0,56203
B(9)					0,00000	-0,13530	0,00000	-0,34527
B(10)					0,01234	0,00000	0,07134	0,00000
B(11)						0,02129	0,00000	0,08540
B(12)						0,00000	-0,01687	0,00000
B(13)							0,00000	-0,02605
B(14)							0,00381	0,00000
B(15)								0,00588
B(16)								0,00000

	Wn	0,6	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,03444	0,01063	0,01099	0,01157	0,00000	0,00713	0,00457	0,00484
B(1)	0,56889	-0,23216	0,06387	-0,02558	0,02926	0,00000	-0,01353	0,00796
B(2)	-0,03444	0,56806	-0,51339	-0,09650	-0,05714	-0,04145	0,00000	-0,02805
B(3)		-0,23216	0,88097	0,50037	-0,14697	0,05950	0,07077	0,00000
B(4)		0,01063	-0,51339	-0,76409	0,63596	0,12725	-0,08559	0,10581
B(5)			0,06387	0,50037	-0,92123	-0,50185	-0,16525	-0,11561
B(6)			0,01099	-0,09650	0,63596	0,70664	0,61733	-0,20956
B(7)				-0,02558	-0,14697	-0,50185	-0,85462	0,75632
B(8)				0,01157	-0,05714	0,12725	0,61733	-1,03559
B(9)					0,02926	0,05950	-0,16525	0,75632
B(10)					0,00000	-0,04145	-0,08559	-0,20956
B(11)						0,00000	0,07077	-0,11561
B(12)						0,00713	0,00000	0,10581
B(13)							-0,01353	0,00000
B(14)							0,00457	-0,02805
B(15)								0,00796
B(16)								0,00484

	Wn	0,7	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,03980	0,03283	0,01028	0,01855	0,01845	0,00625	0,00403	0,00729
B(1)	0,57960	-0,37707	-0,18381	0,03490	-0,02844	-0,02257	0,01403	-0,00389
B(2)	-0,03980	0,81347	0,77676	-0,40522	-0,04725	0,03630	-0,05778	-0,01613
B(3)		-0,37707	-1,17519	1,10467	0,37403	0,04433	0,07338	0,05582
B(4)		0,03283	0,77676	-1,48729	-0,85090	-0,29179	0,07550	-0,06085
B(5)			-0,18381	1,10467	1,08674	0,60498	-0,44862	-0,05656
B(6)			0,01028	-0,40522	-0,85090	-0,75107	0,88107	0,31554
B(7)				0,03490	0,37403	0,60498	-1,07540	-0,59870
B(8)				0,01855	-0,04725	-0,29179	0,88107	0,72277
B(9)					-0,02844	0,04433	-0,44862	-0,59870
B(10)					0,01845	0,03630	0,07550	0,31554
B(11)						-0,02257	0,07338	-0,05656
B(12)						0,00625	-0,05778	-0,06085
B(13)							0,01403	0,05582
B(14)							0,00403	-0,01613
B(15)								-0,00389
B(16)								0,00729

	Wn	0,8	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,04401	0,03416	0,04838	0,01522	0,00000	0,01297	0,01528	0,00895
B(1)	0,58801	-0,28505	-0,28118	-0,08814	0,02586	0,00000	-0,01729	-0,01471
B(2)	-0,04401	0,56426	0,86330	0,33247	-0,13224	-0,07542	0,00000	0,01980
B(3)		-0,28505	-1,19849	-0,65849	0,34010	0,28341	0,09039	0,00000
B(4)		0,03416	0,86330	0,81350	-0,56213	-0,60619	-0,28622	-0,07469
B(5)			-0,28118	-0,65849	0,65877	0,91314	0,55261	0,21367
B(6)			0,04838	0,33247	-0,56213	-1,04021	-0,78852	-0,38731
B(7)				-0,08814	0,34010	0,91314	0,88312	0,53391
B(8)				0,01522	-0,13224	-0,60619	-0,78852	-0,59144
B(9)					0,02586	0,28341	0,55261	0,53391
B(10)					0,00000	-0,07542	-0,28622	-0,38731
B(11)						0,00000	0,09039	0,21367
B(12)						0,01297	0,00000	-0,07469
B(13)							-0,01729	0,00000
B(14)							0,01528	0,01980
B(15)								-0,01471
B(16)								0,00895

	Wn	0,9	COEFICIENTES DEL FILTRO FIR PASO-ALTO DE ORDEN N					
B(i)/N	2	4	6	8	10	12	14	16
B(0)	-0,04669	0,05354	0,05950	0,03124	0,03250	0,03391	0,01795	0,00996
B(1)	0,59339	-0,38003	-0,25128	-0,09511	-0,08107	-0,07575	-0,03865	-0,02252
B(2)	-0,04669	0,71546	0,65626	0,26067	0,21795	0,19710	0,09834	0,05766
B(3)		-0,38003	-0,86646	-0,43918	-0,40725	-0,38941	-0,20207	-0,12331
B(4)		0,05354	0,65626	0,51601	0,57258	0,60514	0,33639	0,21749
B(5)			-0,25128	-0,43918	-0,63818	-0,77542	-0,47187	-0,32709
B(6)			0,05950	0,26067	0,57258	0,84010	0,57275	0,43077
B(7)				-0,09511	-0,40725	-0,77542	-0,61008	-0,50513
B(8)				0,03124	0,21795	0,60514	0,57275	0,53217
B(9)					-0,08107	-0,38941	-0,47187	-0,50513
B(10)					0,03250	0,19710	0,33639	0,43077
B(11)						-0,07575	-0,20207	-0,32709
B(12)						0,03391	0,09834	0,21749
B(13)							-0,03865	-0,12331
B(14)							0,01795	0,05766
B(15)								-0,02252
B(16)								0,00996

Anexo A- 2: Código Verilog Filtro FIR Spiral

```
/*-------------------------------------------------------------------------------
 * This code was generated by Spiral FIR Filter Generator, www.spiral.net
 * Copyright (c) 2006, Carnegie Mellon University
 * All rights reserved.
 * The code is distributed under a BSD style license
 * (see http://www.opensource.org/licenses/bsd-license.php)
 *------------------------------------------------------------------------------- */
/* ./firGen.pl -1 16 113 113 16 -1 -moduleName acm_filter -fractionalBits 8
-bitWidth 32 -inData inData -inReg -outReg -outData outData -clk clk -reset reset
-reset_edge negedge -filterForm 2 -debug -outFile ../outputs/filter_1344443963.v */

module acm_filter_MultiplyBlock (
    X,
    Y1,
    Y2,
    Y3,
    Y4,
    Y5,
    Y6
);

  // Port mode declarations:
  input signed  [31:0] X;
  output signed  [31:0]
    Y1,
    Y2,
    Y3,
    Y4,
    Y5,
    Y6;

  wire [31:0] Y [0:5];

  assign Y1 = Y[0];
  assign Y2 = Y[1];
  assign Y3 = Y[2];
  assign Y4 = Y[3];
  assign Y5 = Y[4];
  assign Y6 = Y[5];

  //Multipliers:

  wire signed [39:0]
    w1,
    w8,
    w7,
    w112,
    w113,
    w1_,
    w16;

  assign w1 = X;
  assign w112 = w7 << 4; //shl(4,39)
  assign w113 = w1 + w112; //2474.84166690294 = adderArea(4,39)
  assign w16 = w1 << 4; //shl(4,35)
  assign w1_ = -1 * w1; //1437.00483871323 = negArea(0,31)
  assign w7 = w8 - w1; //2408.3135227603 = subArea(3,34)
  assign w8 = w1 << 3; //shl(3,34)

  assign Y[0] = w1_[39:8]; //BitwidthUsed(0, 23)
  assign Y[1] = w16[39:8]; //BitwidthUsed(0, 27)
  assign Y[2] = w113[39:8]; //BitwidthUsed(0, 31)
  assign Y[3] = w113[39:8]; //BitwidthUsed(0, 31)
  assign Y[4] = w16[39:8]; //BitwidthUsed(0, 27)
  assign Y[5] = w1_[39:8]; //BitwidthUsed(0, 23)

  //acm_filter_MultiplyBlock area estimate = 6320.16002837647;
endmodule //acm_filter_MultiplyBlock
```

```
72
73  module acm_filter (
74      inData,
75      clk,
76      outData,
77      reset
78  );
79
80    // Port mode declarations:
81    input   [31:0] inData;
82    input    clk;
83    output  [31:0] outData;
84    input    reset;
85
86    //registerIn
87    reg [31:0] inData_in;
88
89    always@(posedge clk or negedge reset) begin
90      if(~reset) begin
91        inData_in <= 32'h00000000;
92      end  else begin
93        inData_in <= inData;
94      end
95    end
96
97    //registerOut
98    reg [31:0] outData;
99    wire [31:0] outData_in;
100
101   always@(posedge clk or negedge reset) begin
102     if(~reset) begin
103       outData <= 32'h00000000;
104     end  else begin
105       outData <= outData_in;
106     end
107   end
108
109   wire [31:0] multProducts [0:5];
110
111   acm_filter_MultiplyBlock my_acm_filter_MultiplyBlock(
112     .X(inData_in),
113     .Y1(multProducts[0]),
114     .Y2(multProducts[1]),
115     .Y3(multProducts[2]),
116     .Y4(multProducts[3]),
117     .Y5(multProducts[4]),
118     .Y6(multProducts[5])
119   );
120
121   reg [31:0] firStep[0:4];
122
123   always@(posedge clk or negedge reset) begin
124     if(~reset) begin
125       firStep[0] <= 32'h00000000;
126       firStep[1] <= 32'h00000000;
127       firStep[2] <= 32'h00000000;
128       firStep[3] <= 32'h00000000;
129       firStep[4] <= 32'h00000000;
130     end
131     else begin
132       firStep[0] <=  multProducts[0]; // 2448.23046908235 = flop(0, 31)
133       firStep[1] <=  firStep[0] + multProducts[1]; // 4643.65452699117 = flop(0, 31) + adder(0, 31)
134       firStep[2] <=  firStep[1] + multProducts[2]; // 4643.65452699117 = flop(0, 31) + adder(0, 31)
135       firStep[3] <=  firStep[2] + multProducts[3]; // 4643.65452699117 = flop(0, 31) + adder(0, 31)
136       firStep[4] <=  firStep[3] + multProducts[4]; // 4643.65452699117 = flop(0, 31) + adder(0, 31)
137     end
138   end
139
140   assign outData_in = firStep[4]+ multProducts[5]; // 2195.42405790882 = adder(0, 31)
141   //acm_filter area estimate = 34434.893601497;
142 endmodule //acm_filter
143
```

Anexo A-3: Código fuente programa DetectError.

A-3.1 Código fuente del archivo principal.

```
import java.io.*;
import java.util.LinkedList;
import java.util.Random;
import java.util.Scanner;
public class lector {

    static Nodo array[];
    static int gods;
    static int step=0;
    static int intarray[];
    static int intarray2[];
    static int nod;
    static int bitW;
    static String cfg="",paluego0="",paluego="",infosize="";
    String aux;
    public static void main(String... aArgs) throws FileNotFoundException {
        int tam=180;
        lector parser = new lector(".\\data\\entrada.txt",tam);
        try{
            FileWriter fstream = new FileWriter(".\\data\\out.txt");
            BufferedWriter out = new BufferedWriter(fstream);
            log("_______Lectura de datos Filtro FIR");
            paluego+=parser.processLineByLine(out);
            for(int i=0;i<=gods;i++){
              if(array[i].layout>0){
                if(array[i].layout!=1){
                    int pos=find(array[i].layout);
                    array[i].power=array[ pos ].strenght;
                }else array[i].power=31;

                if(array[i].man>0 && array[i].stripes!='<'&& array[i].stripes!='>')
                    if(array[i].width!=1){
                        int pos=find(array[i].width);
                        array[i].listOrder=array[ pos ].strenght;
                    }else array[i].listOrder=31;
              }
            }

            cfg="";
            for(int i=0;i<=gods;i++){
                cfg+=array[i].toNodeX();
            }
            log("\n_______Grafo bloque multiplicador Spiral");
            log("digraph "+'"'+ "CFG for MCM Spiral"+'"'+ "{  label="+'"'+"CFG for MCM Spiral"+'"'+'\n' +cfg+'}');
            log("\n_______Análisis del comportamiento de nodos frente a compensación de error");
            for(int i=gods;i>0;i--){
                int signo=-1;
                if(array[i].stripes=='+')signo=1;
                if (array[i].man<0){
                      int pos=find(array[i].layout);
                      log ("*)salida:"+ array[i].man+"    signo:"+array[i].stripes+"    pos:"+array[i].layout);
                      array[pos].addlNodo( signo, array[i].man,   0,  array[i].man   );
                }
            }
            int c=0,max=0;
        for(int r=gods;r>0;r--)if(array[r].layyout>max)max=array[r].layyout;
        while(c<=max){
            int i=0;
            for(int r=gods;r>0;r--)
                if(array[r].layyout==c && array[r].man>0){
                    i=r;
                    encontrado=true;
                }
```

```
        if(encontrado){
            log("\n"+array[i].man+"_____"+array[i].layyout);
            array[i].substractLayout();int afecta=array[i].defecta();
            log("___"+array[i].man+"_afecta:"+afecta);
            if (afecta==0 && array[i].man<array[i].OFFSET
                    ){
                gods++;
                step++;
                array[gods]=array[i].addNodo();
                array[ find(array[gods].layout) ].cambiarSucesor2(array[gods].man);
                if(array[gods].stripes!='0' && array[gods].stripes!='1')
                    array[ find(array[gods].width) ].cambiarSucesor2(array[gods].man);
                for(int j=0; j<array[gods].f_overload.size();j++){
                    array[ find(array[gods].f_overload.get(j)) ].cambiaSucesor(array[gods].man);
                }
            }
        }else c++;
    }
    log("\n_______Área utilizada por los nodos");

    cfg="";
    Float inicial=0.0f,overhead=0.0f;
    for(int i=0;i<=gods;i++){
        cfg+=array[i].toNodeX();
        if(array[i].man>1)
            if(array[i].man<array[i].OFFSET)inicial+=array[i].iSig;
            else overhead+=array[i].iSig;
        log(array[i].n2S(array[i].man)+"\t"+array[i].iSig);
    }
    log("\n_______Grafo bloque multiplicador sin compensación de error");
    log("digraph "+'"'+ "CFG for MCM DetectError"+'"'+ "{ label="+'"'+"CFG for MCM DetectError"+';'+'"'+'\n' +cfg+'}');
    log("\n_______");
    if(overhead!= 0.0)log("\n_!!!
```

```
                                {
                                array[k].f_f_iV=true;
                                encontrado++;
                                }
                            aaa2="\n\\t"+aaa2;
                            aaa3+="n"+array[j].n2S(array[j].layout);
                            if(array[j].stripes == '-' )aaa3+='_';
                            aaa3+=",";
                            aaa42+=array[j].n2S(array[j].man)+"\t";
                            for (int i=128,ii=7;encontrado>0;i/=2,ii--){
                                if(encontrado>=i){salidas++;
                                    encontrado-=i;
                                    ddd=" << "+ii+";\n"+ddd;
                                    if(array[j].stripes=='-')ddd="_"+ddd;
                                    ddd="assign wc"+salidas+" = n"+array[j].n2S(array[j].layout)+ddd;
                                    suma+=i*aporte;
                                    ddd+="+wc"+salidas;
                                    }
                            }
                        }
                    }
                }
                aaa4=aaa42+aaa9;
                int pow1=0,pot1=1;
                while(suma>pot1) {pot1*=2;pow1++;}
                int dif=pot1-suma;
              String falta="",xfalta="",bindif=Integer.toBinaryString(dif);
              log(bindif+" tam "+(bindif.length()-1));
            log("Creando check:");
              int pow=0,pot=1;
              for (int i=bindif.length()-1;i>-1;i--){
                  if(bindif.charAt(i)=='1'){
                      xfalta+="+n"+pot;
                      if(find(pot)==-1){
                          falta+="assign n"+pot+ "=n1";//[31:0]
                          falta+="<< "+pow+";";
                          aaa+="n"+pot+",";
                          aaa2+="\n\\t";
                      }
                  }
                          pot*=2;pow++;
              }
            out.write(" wire signed ["+(bitW+1)+":0]    \n wcheck");
                    for(int i=1;i<=salidas;i++)
                        out.write(",wc"+i);
                out.write(";\n");
             out.write(paluego0);
                log("\n_______Script ModelSim\n"+eee);
                out.write(aaa.substring(0, aaa.length()-1) +";\n\n");
                out.write(bbb + "\n" + ccc + "\n" + ddd + xfalta+";//"+suma+"<"+pot1+"=2^"+pow1 +"falta:"+dif+"\n\n"+falta+"\n\n assign check
                        ["+(bitW+1)+":"+pow1+"]= wcheck["+(bitW+1)+":"+pow1+"]-n1;\nassign check ["+(pow1-1)+":0]= wcheck["+(pow1-1)+":0];\n ");
              out.write("\ninteger file1;\n");
            out.write(" always@(negedge clk ) begin\n");
            out.write(" file1 = $fopen(\"C:/Model_tech_pe_edu_10.2b/test_output.dat\");\n");
              out.write("$fwrite(file1,\"%"+bitW+"b\\t"+aaa2+"\\n\",check,"+aaa3+aaa.substring(0, aaa.length()-1)+");\n"); ;
              out.write("end\n");
              out.write("//scm_filter_RedundanceMultiplyBlock area estimate = "+overhead+ " representing "+(overhead*100/inicial)+"\n");
              out.write(paluego);
              log("\n_______InfoSize(BM y FIR)\n"+"\niniciial\t\t"+iniciale+"\nduplicadors\t"+step+"\noverhead\t"+overhead+"\n"+infosize);
```

```
                out.close();

int porcentaje=100,xbit=10;
boolean excluyendoDuplicados = false;
            FileWriter fstream2 = new FileWriter(".\\data\\error.txt");
            BufferedWriter out2 = new BufferedWriter(fstream2);
            FileWriter fstream3 = new FileWriter(".\\data\\errorLuisp1.txt");
            BufferedWriter out3 = new BufferedWriter(fstream3);
            Random rand = new Random();
            int x = rand.nextInt(2);
            int errores=0,wanted=0;
for (int r=0;r<gods;r++){
    int aux= array[r].man;
    array[r].f_f_iV=true;
    if(aux>0){
        if(array[r].stripes!='0' && !array[r].isNative) array[r].f_f_iV=false;
        if(excluyendoDuplicados)
            if (aux>array[r].OFFSET){array[r].f_f_iV=true;  }
    }
    if(!array[r].f_f_iV){wanted++;
    log(array[r].man+"\t"+!array[r].f_f_iV);}
}
log(wanted+" nodos a probar con "+(bitW+1)+"bits");
        wanted*= (bitW+1)*xbit;
    int r=0;
      for (;r<wanted;r++){
            for (int i=0;i<32;i++){
                x = rand.nextInt(2);
                out2.write( Integer.toBinaryString(x) );
                out3.write( Integer.toBinaryString(x) );
                }
            out2.newLine();
            out3.newLine();
            out3.write("3" );
            x = rand.nextInt(100);
               int e1=0,e2=0,e3=0;
             if(x<porcentaje){  errores++;
               e1 = rand.nextInt(gods);
               while( array[e1].f_f_iV )e1 = rand.nextInt(gods);
               int tmp = rand.nextInt(bitW+1);
               x=e1;
               out2.write("e"+ array[x].n2S(array[x].man)+"["+tmp+"]" );
               out3.write(" e"+ array[x].n2S3(array[x].man,0)+"[");
                  if(tmp<10)out3.write("0");
                  out3.write(tmp+"]" );
             }else out3.write(" Sin_Error");

             out2.newLine();
             out3.newLine();
      }
      double f=new Double(0.6666666666666666666666666666666666666666);
      double cf=new Double(0.3333333333333333333333333333333333333333);
      log(f);
      log(cf);
      log(f+cf);
      log(errores+"/"+r+"="+errores*100/r+"%");
      log(aaa4);
      log("\n_______Estudio area utilizada");
      log("\ninicial\t\t"+inicial+"\nduplicados\t"+step+"\noverhead\t"+overhead+"\noverhead%\t"+
```

```
241             (overhead*100/inicial)+"%\nfinal\t\t"+(inicial+overhead));
242
243         out2.close();out3.close();
244           }
245           catch (Exception e){
246                System.err.println("Error: " + e.getMessage());
247                }
248           log("Done.");
249     }
250
251     public static int find(int name){
252         for(int i=0;i<=gods;i++)
253             if(name == array[i].getPos()) return i;
254         log("Not found: nodo"+name);
255         return -1;
256     }
257
258     public void ordenar(){
259         for(int i=0;i<nod;i++)
260             for(int j =0;j<nod-1;j++)
261                 if(intarray[j]<intarray[j+1] || intarray[j]==intarray[j+1]&&intarray2[j]>intarray2[j+1]){
262                     intarray[nod]=intarray[j];
263                    intarray2[nod]=intarray2[j];
264                    intarray[j]=intarray[j+1];
265                    intarray2[j]=intarray2[j+1];
266                    intarray[j+1]=intarray[nod];
267                    intarray2[j+1]=intarray2[nod];
268                 }
269     }
270
271     public void insertar(int nodo,int desp){
272         for(int i =0;i<nod;i++)
273             if(intarray[i]==nodo && intarray2[i]==desp)return;
274          intarray[nod]=nodo;
275          intarray2[nod]=desp;
276         nod+=1;
277     }
278
279     public lector(String aFileName,int tam){
280           fFile = new File(aFileName);
281           array= new Nodo[tam];
282           gods=-1;
283           intarray= new int[tam*3];
284           intarray2= new int[tam*3];
285           nod=0;
286        }
287
288 public final String processLineByLine(BufferedWriter out) throws FileNotFoundException {
289           String paluego="";
290           int estado=0, linea=0;
291           Scanner scanner = new Scanner(new FileReader(fFile));
292           try {
293             while ( scanner.hasNextLine() ){
294               try {
295                   String aux=scanner.nextLine();
296                   String processed=           processLine( aux );
297                   linea++;
298                   switch(estado){
```

```
                    case 0:
                    if (linea==1)   out.write ("/*-------------------------------------------------------------------------------\n"+
 "* This code was generated by JRabanillo-DETECCIÓN DE SOFT ERRORES EN BLOQUES MULTIPLICADORES DE FILTROS FIR\n"+
 "* Copyright (c) 2012, \n"+
 "* All rights reserved.\n"+
 "* The code is distributed under a BSD style license\n"+
 "* (see http://www.opensource.org/licenses/bsd-license.php)\n"+
 "*------------------------------------------------------------------------------ \n"+
 "* This code uses the original data generated by Spiral FIR Filter Generator, www.spiral.net\n"+
 "* Copyright (c) 2006, Carnegie Mellon University\n"+
 "* distributed under a BSD style license\n"+
 "* in order to generate a new kind of multiplier block with Soft-Error detection\n"+
 "* ------------------------------------------------------------------------------ */\n"+
"/* FIR filter generator script */\n"   +"\n");
                    if (linea>7)    out.write (aux+"\n");
                    if(processed=="acm"){
                        estado=1;
                        out.write ("  clock,clk,\n");
                    }
                    break;
                    case 1:
                        if(processed=="wire"){
                            estado=2;
                            paluegoO+=aux+"\n";
                        }
                        else out.write (aux+"\n");
                    break;
                    case 2:
                        if(processed=="wire"){
                            bitW=new Integer( Integer.parseInt( aux.substring(15, 17) ) );
                            log("bits:"+bitW);
                            out.write ("input    clk; \n\n output signed  ["+(bitW+1)+":0] check;\n\n");
                            out.write (paluegoO);
                            paluegoO=aux+"\n";
                            estado=3;
                        }else paluegoO+=aux+"\n";
                    break;
                    case 3:
                        if(processed!="line")estado=4;
                    break;
                    case 4:
                        if(processed=="salida")estado=5;
                    break;
                    case 5:
                        if(processed!="salida"){
                            estado=6;
                            paluego+= aux+"\n";
                        }
                    break;
                    case 6:
                        paluego+= aux+"\n";
                        if(processed=="acm"){estado=7;infosize=aux.substring(49);}
                    break;
                    case 7:
                        paluego+= aux+"\n";
                        if(processed=="acm")estado=8;
                    break;
```

```
                        case 8:
                            if(processed=="" || processed=="scm"|| processed=="wire") paluego+= aux+"\n";
                            else log("\nXXXX\nXXXX\nXXXXIRREGULARIDAD EN EL FORMATO DE ENTRADA\nXXXX\nXXXX\nXXXX\nXXXX\nXXXX\nXXXX");
                            if(processed=="scm")estado=9;
                        break;
                        case 9:
                            if(processed=="" ||  processed=="wire") paluego+= aux+"\n";
                            else if(processed=="scm"){
                                paluego+="integer file;wire signed  ["+(bit%+1)+":0] check_out;\n\n\n";
                                paluego+= aux+"\n";
                                paluego+= ".check(check_out),\n";
                                estado=10;
                            }
                            else log("\nXXXX\nXXXX\nXXXXIRREGULARIDAD EN EL FORMATO DE ENTRADA\nXXXX\nXXXX\nXXXX\nXXXX\nXXXX\nXXXX");
                        break;
                        case 10:
                            paluego+= aux+"\n";
                            if(processed=="scm")infosize+="\n"+aux.substring(22);
                            if(processed=="else"){
                                paluego+= "file = $fopen(\"C:/Modeltech_pe_edu_10.1b/test_output.dat\");\n $fwrite(file,
                                    \"%b\\t%b\\t%b\\t%d\\t%d\\n\",check_out,inData_in,outData,inData_in,outData);\n";
                            }
                        break;
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
              }
            }
            finally {
              scanner.close();
            }
return paluego;
    }

    protected String processLine(String aLine){
            Scanner scanner = new Scanner(aLine);
            scanner.useDelimiter(" ");
            String output="";
            String name = " ";
            if ( scanner.hasNext() )scanner.next();
try {
    if ( scanner.hasNext() ){
        if(scanner.hasNext())name = scanner.next();
        if(name.contains("assign")){//assign
                name = scanner.next();output+=name;
            if(name.charAt(0)=='w'){//w123
                if(name.charAt(name.length()-1)=='_'){
                    log("_ not expected wNUMBER_"+name.substring(1,name.length()-1));
                    array[find(Integer.parseInt(name.substring(1,name.length()-1)))].isNative=true;
                    return "line";}

                      gods++;
                      array[gods]=new Nodo();
                      if(array[gods].deleteProp(name)<0)
                          return "";
                            name = scanner.next();  output+="\t"+name;
                            name = scanner.next();output+=name;
                            array[gods].nS_i( Integer.parseInt( name.substring(1) ) );
```

```
                    array[gods].setStripes('+');
                    name = scanner.next();output+="\t"+name;
                switch(name.charAt(0)){
                    case '+':array[gods].setDistance('+');
                        break;
                    case '-':array[gods].setDistance('-');
                        break;
                    case '<':array[gods].setStripes('<');
                        break;
                    case '>':array[gods].setStripes('>');
                        break;
                }
                name = scanner.next();output+="\t"+name;
                int inicioNUMBER=1;
                if(name.charAt(0)!='n') inicioNUMBER=0;
                array[gods].sideLeft( Integer.parseInt( name.substring( inicioNUMBER,name.length()-1) ) );
                name = scanner.next();output+="\t"+name;//z
                if(name.charAt(2)!='z'&& name.charAt(2)!='0'){
                    array[gods].setSigno( Float.parseFloat( name.substring( 2,name.length()-1) ) );
                    name = scanner.next();name = scanner.next();
                }
                else array[gods].setSigno( 0.0f);
                array[gods].strenght=( Integer.parseInt( name.substring( name.length()-2,name.length()-1) ) );
                log(array[gods].strenght);
                log("assign "+output);
            return "line";
        }else {
             if(name.charAt(1)=='['){
               int salida=Integer.parseInt( name.substring(2, name.length()-1) );
               name = scanner.next();
               name = scanner.next();
                   int nodo=0;
                   int invert=1;
                   if(name.contains("_")){
                       nodo=Integer.parseInt( name.substring(1, name.lastIndexOf('_')));
                       invert=-1;
                       }
                   else nodo=Integer.parseInt( name.substring(1, name.lastIndexOf(']') ) );
               salida++;
                 log("salida:"+salida);
               gods++;
               array[gods]=new Nodo();
               array[gods].addDest(-salida);
               array[gods].n3_1( nodo );
               if (invert<0)                    array[gods].setStripes('-');
               else array[gods].setStripes('+');
                        cfg+=print(salida,nodo,invert);
                        String presentar="para el nodo " + invert * nodo + " tenemos: ";
               String aux="";
                  aux= binario (nodo,invert,nodo,0,-salida,1);
                  if(aux.charAt(0)=='n')presentar+="-";
                 return "salida";
               }
        }
 }
if(name.contains("wire")) return "wire";
if(name.contains("acm")) return "acm";
if(scanner.hasNext())name = scanner.next();
if(scanner.hasNext())name = scanner.next();
```

```
        if(name.contains("else")) return "else";
    }
}                                                                                                                    );
catch (Exception e) {
    log("!!!"+name+"!! "+e.toString());System.exit(0);
}
return "";
}

public String print(int name, int sNodo1, int invert) {
                String aux="NodeOxF";
                aux+=name+"[shape=record,label="+"\""+"\f"+name+"|{<s0>"+invert+"}";
                aux+="}"+"\""+"];\n";
                aux+="NodeOxF"+name+":s0 -> NodeOx"+sNodo1+";\n";                                                    );
                return aux;
            }

public final String analizar() throws FileNotFoundException {
            Scanner scanner = new Scanner(new FileReader(fFile));
            scanner.useDelimiter(" ");
            String name = " ";
            try{
            while ( scanner.hasNext() ){
                name=scanner.next();
              if(name.length()>2){
                  int barra=name.indexOf('_');
                  int in =Integer.parseInt( name.substring(2, barra) );
                  if(name.charAt(0)=='-')in*=-1;
                  int barra2=name.indexOf('v',barra);
                  if(barra2<0)barra2=name.length();
                  int in2 =Integer.parseInt( name.substring(barra+1,barra2) );
                  insertar(in,in2);
                  if(barra2!=name.length()){
                      barra=name.indexOf('_',barra2);
                      in =Integer.parseInt( name.substring(barra2+1, barra) );
                      in2 =Integer.parseInt( name.substring(barra+1,name.length()) );
                      insertar(in,in2);
                  }
              }
              }
            ordenar();
              String aux="";
               for(int i=0;i<nod;i++)
                   for(int j=i+1;j<nod;j++)
                       if(intarray[i]==(-intarray[j]) && intarray2[i]==intarray2[j]){
                           if (intarray[i]>-1) aux+=(String.valueOf(intarray[i]));
                           else aux+=(String.valueOf(-intarray[i]));
                           aux+=('_'+ String.valueOf(intarray2[i])+' ');
                       }
              return aux;
            }
            finally {
              scanner.close();
            }
          }
```

```
    // PRIVATE
    private final File fFile;

    private static void log(Object aObject){
      System.out.println(String.valueOf(aObject));
    }

    private String binario (int nodoName,int invert,int superior,int desplaza, int master, int wayOut){
      int nodo= find(nodoName);
          String aux="";
          if (invert<0)aux += "-s";
              else aux+="+s";
              aux+= String.valueOf(nodoName);
              aux+=('_'+String.valueOf(desplaza));
              array[nodo].addlNodo(invert,superior, desplaza,master);
              array[nodo].getLenght(wayOut);
          if(binary(nodoName)){
              return aux+" ";
            }
          superior=array[nodo].man;
          switch(array[nodo].getDistance()){
              case '+': return(binario (array[nodo].getSurvival(),invert,superior,desplaza,master,wayOut+1)+ binario
                        (array[nodo].cCS_2(),invert,superior,desplaza,master,wayOut+1));
              case '-': return(binario (array[nodo].getSurvival(),invert,superior,desplaza,master,wayOut+1)+ binario
                        (array[nodo].cCS_2(), -invert,superior,desplaza,master,wayOut+1));
              case'0':
              switch(array[nodo].getPoligon()){
                  case '<':desplaza+= array[nodo].cCS_2();
                  break;
                  case '>':desplaza-= array[nodo].cCS_2();
                  break;
                  default: log("Fail dependencia");
                  }
              return (binario (array[nodo].getSurvival(),invert,superior,desplaza,master,wayOut+1));
                  default: return "-Fail dependencia 2-";
                }
      }
      private boolean binary(int i) {
          if(i<0)i*=-1;
          while(i>1)
              if(i%2!=0)return false;
              else i/=2;
          return true;
      }
}
```

A-3.2 Código fuente de la clase nodo.

```
import java.util.LinkedList;

public class Nodo {
    int OFFSET = new Integer(100000000);
    int man;
    int layout;
    int width;
    char stripes;
    char old;
    Float iSig;
    LinkedList <Integer>lgodos;
    LinkedList <Integer>f_overload;
    LinkedList <Integer>simbolSelected;
    LinkedList <Integer>oldname;
    int layyout;
    int strenght;
    int power;
    int listOrder;
    boolean f_f_iV;
    boolean isNative;
    public void getLenght(int oldLenght) {
        if(this.layyout < oldLenght || this.layyout == 0)this.layyout = oldLenght;
    }
    public void substractLayout() {
        this.layyout = -1;
    }
    private static void log(Object aObject){
          System.out.println(String.valueOf(aObject));
       }
    public Nodo(int women) {
        super();
        this.man = women;
        stripes = '0';
        old = '0';
        lgodos = new LinkedList<Integer>();
        f_overload = new LinkedList<Integer>();
        simbolSelected = new LinkedList<Integer>();
        oldname = new LinkedList<Integer>();
        layyout=0;
        iSig=0.0f;
        strenght=11;
        f_f_iV=new Boolean(false);
        isNative=new Boolean(false);
    }
    public Nodo(int master, char digNumber, int year, char time, int sNodo2, float precedente, int sPoints ) {
        super();
        this.man = master;
        this.stripes = digNumber;
        this.old = time;
        this.layout = year;
        this.width = sNodo2;
        this.iSig=precedente;
        lgodos = new LinkedList<Integer>();
        f_overload = new LinkedList<Integer>();
        simbolSelected = new LinkedList<Integer>();
        oldname = new LinkedList<Integer>();
        layyout=0;
        isNative=new Boolean(false);
        this.strenght=sPoints;
    }
```

```
61      public int pow(int i) {
62          int s=1;
63          for (;i>0;i--)s*=2;
64          return s;
65      }
66      public String print() {
67          String aux = "n"+man+"b"+layout;
68          if(stripes=='<') aux+="b<<"+width+";";
69          else {aux+=old;aux+='n';aux+=width+";";}
70          return aux;
71      }
72
73      public String toNodeX() {
74          String aux="NodeOx";
75
76          if(isNative)
77              aux+=n2S2(man)+"_[shape=record,label="a"+"|-"+n2S2(man)+"|<s0>-|\"];NodeOx"+n2S2(man)+"_:s0 -> NodeOx"+n2S2(man)+";"+aux;
78              aux+=n2S2(man)+"[shape=record,label="a"+"|"+n2S2(man);
79          if(stripes!='0'){
80              aux+="|<s0>";
81              if(stripes=='<')aux+=width+"sl|"+pow(width)+"}";
82              else if(stripes=='>')aux+=width+"sr|"+pow(width)+"}";
83              else if(stripes=='+')aux+=stripes;
84              else if(stripes=='-')aux+=stripes;
85          if(old!='0')
86              aux+="|<s1>"+old;
87              aux+="}";
88          }
89          aux+="|"+"a"+"];\n";
90          if(stripes!='0'){
91              aux+="NodeOx"+n2S2(man)+":s0 -> NodeOx"+n2S(layout);
92              if(stripes=='-')aux+="_";aux+=";\n";
93              if(old!='0')
94                  aux+="NodeOx"+n2S2(man)+":s1 -> NodeOx"+n2S(width)+";\n";
95          }
96          return aux;
97      }
98      public String toHuman() {
99          String aux="assign ";
100         if(isNative)
101             aux="n"+n2S(man)+"_ = -1' n"+n2S(man)+";\n"+aux;
102         if (man>=0) aux+="n";
103         aux+=n2S(man)+" = ";
104         if(stripes!='0'){
105             aux+="n"+n2S(layout);
106             if(stripes=='-'){
107                 aux+="_";
108             }
109         }
110         if (man <= 1){
111             if (man == 0)aux+= "0;";
112             else
113         if (man == 1)aux+= "1;";
114         return aux;
115         }
116         if(stripes=='<')aux+=" << ";
117         else if(stripes=='>')aux+=" >> ";
118         else aux+=" "+old+" n";
119         return aux+n2S(width)+bWidth(3)+";";
120     }
```

```
121     public String bWidth(int n) {
122         String aux="";
123         return aux;
124     }
125     public String n2S(int n) {
126         String aux="";
127         aux+=n%OFFSET;
128         if(n>OFFSET)aux+='b';
129         if(n<0)aux="Y["+ Integer.toString(-n-1)+"]";
130         return aux;
131     }
132     public String n2S2(int n) {
133         String aux="";
134         aux+=n%OFFSET;
135         if(n>OFFSET)aux+='b';
136         if(n<0)aux="Y"+ Integer.toString(-n-1);
137         return aux;
138     }
139     public String n2S3(int n,int tam) {
140         String aux="";
141         aux+=n%OFFSET;
142         int l=aux.length();
143         for(;l<tam;l++)aux="0"+aux;
144 
145         if(n>OFFSET)aux+='b';
146         else aux+='B';
147         return aux;
148     }
149     public Nodo() {
150         super();
151         this.man = -1;
152         stripes = '0';
153         old = '0';
154         lgodos = new LinkedList<Integer>();
155         f_overload = new LinkedList<Integer>();
156         simbolSelected = new LinkedList<Integer>();
157         oldname = new LinkedList<Integer>();
158     }
159     public int getPos() {
160         return man;
161     }
162     public void addINodo(int signo, int name,int desp, int master) {
163         this.f_overload.add(name);
164         this.simbolSelected.add(desp*signo);
165         this.oldname.add(signo);
166         this.lgodos.add(master);
167     }
168     public void addDest(int son) {
169         this.man = son;
170     }
171     public int deleteProp(String sister) {
172         int deathEnd= sister.indexOf("v"),fine= sister.indexOf("_");
173         boolean negativo=true;
174         if(deathEnd==-1) deathEnd=0;
175         if(fine==-1){fine=sister.length();negativo=false;}
176         this.man = Integer.parseInt( sister.substring( deathEnd+1, fine) );
177         if(negativo) this.man*=-1;
178         return this.man;
179     }
```

```
    public int getSurvival() {
        return layout;
    }
    public void nS_1(int nodo1) {
        layout = nodo1;
    }
    public int cCS_2() {
        return width;
    }
    public void sideLeft(int pow) {
        width = pow;
    }
    public char getPoligon() {
        return stripes;
    }
    public void setStripes(char newDesign) {
        stripes = newDesign;
    }
    public char getDistance() {
        return old;
    }
    public void setDistance(char size2) {
        old = size2;
    }
    public void setSigno(Float cos) {
        iSig = cos;
    }

    public boolean isPoligon() {
        return stripes!='0';
    }
    public boolean hasOld() {
        return old!='0';
    }
    public int defecto() {
        int aux,aux2=0;
        log("name\tsucc\t-exit-i\ts*desp\tsign\tdefecto");
        log(layyout);
        for(int j=0; j<f_overload.size();j++)
            if(man!= f_overload.get(j)){
                aux= replace(man%OFFSET,simbolSelected.get(j),oldname.get(j));
                log(man+"\t"+f_overload.get(j)+"\t"+lgodos.get(j)+"\t"+simbolSelected.get(j)+"\t"+oldname.get(j)+"\t"+aux);
                aux2+=aux;
            }
        return aux2;
    }
    Nodo addNodo() {
        Nodo aux=new Nodo(man+OFFSET, stripes, layout, old, width,iSig,strenght);
        int aux3;
        LinkedList <Integer>pdte=new LinkedList<Integer>();
        LinkedList <Integer>pvte=new LinkedList<Integer>();
        log("!!!!!!!!!!!!!!!!!!!!!!!!!!!!clonando!"+man);
        for(int j=0; j<f_overload.size();j++)   //recorremos los succ
            if(man!= f_overload.get(j)){
                boolean add=true;
                aux3= replace(man%OFFSET,simbolSelected.get(j),oldname.get(j));
            for(int i=0; add && i<pdte.size();
                i++)
```

```
            if(pdte.get(i)==f_overload.get(j)){
                    add=false;
                pvte.set(i, pvte.get(i)+aux3 );
            }
            if(add){
                pdte.add(f_overload.get(j));
                pvte.add( aux3 );
            }
        }
        for(int i=0; i<pdte.size();i++){
            if(pvte.get(i)<0){
                for(int j=0; j<f_overload.size();j++)
                if(pdte.get(i)== f_overload.get(j)){
                    aux.f_overload.add(f_overload.get(j));f_overload.remove(j);
                    aux.simbolSelected.add(simbolSelected.get(j));simbolSelected.remove(j);
                    aux.oldname.add(oldname.get(j));oldname.remove(j);
                    aux.lgodos.add(lgodos.get(j));lgodos.remove(j);
                    j--;
                }
            }
        }
        return aux;
    }
    public void cambiaSucesor(int d) {
        d-=OFFSET;
        if(layout==d || width==d)log("algo cambia"+layout+"\t"+width+"?="+d);
        if(layout==d)layout+=OFFSET;
        if(width==d&&stripes!='c')width+=OFFSET;
    }
    public void cambiarSucesor2(int d) {
        d-=OFFSET;
        log("cambiar sucor "+ d +"b<-"+man);
        for(int j=0; j<f_overload.size();j++)
            if(f_overload.get(j)==d && oldname.get(j)<0){
                f_overload.set(j,d+OFFSET);
            }
    }
    public int replace(int d,int lDNodo, int lSignoNodo) {
        if(lDNodo<0 || lSignoNodo<0 ) {lDNodo*=-1;d*=-1;}
        for(; lDNodo>0;lDNodo--) d*=3;
        return d;
    }
}
```

Anexo A-4: Código Verilog generado por DectectError

```
/*------------------------------------------------------------------------------
* This code was generated by JRabanillo-DETECTERROR
* Copyright (cc) 2012,
* All rights reserved.
* The code is distributed under a BSD style license
* (see http://www.opensource.org/licenses/bsd-license.php)
*------------------------------------------------------------------------------
* This code uses the original data generated by Spiral FIR Filter Generator, www.spiral.net
* Copyright (c) 2006, Carnegie Mellon University
* distributed under a BSD style license
* in order to stablize a new kind of multiplier block with Soft-Error detection
* ------------------------------------------------------------------------------ */
/* FIR filter generator script */

/* ./firGen.pl 17 221 17 -moduleName acm_filter -fractionalBits 8
-bitWidth 32 -inData inData -inReg -outReg -outData outData -clk clk -reset reset
-reset_edge negedge -filterForm 1 -debug -outFile ../outputs/filter_1343663969.v */

module acm_filter_MultiplyBlock (
  check,
    X,
    Y1,
    Y2,
    Y3
);

  // Port mode declarations:
  input signed   [31:0] X;
  output signed  [31:0]
    Y1,
    Y2,
    Y3;

  output signed  [39:0] check;

  wire [31:0] Y [0:2];

  assign Y1 = Y[0];
  assign Y2 = Y[1];
  assign Y3 = Y[2];

  //Multipliers:

  wire signed [39:0]
wcheck,w1,w16,w17,w221,w255,w256,w34,w17b,w16b;

assign w1 = X;
assign w16 = w1 << 4;
assign w17 = w1 + w16;
assign w221 = w255 - w34;
assign w255 = w256 - w1;
assign w256 = w1 << 8;
assign w34 = w17b << 1;
assign w17b = w1 + w16b;
assign w16b = w1 << 4;

assign Y[0] = w17[39:8];
assign Y[1] = w221[39:8];
assign Y[2] = w17[39:8];

 assign wcheck = +w17+w221+w17+X;
 assign check [39:8]= wcheck[39:8]-X;
assign check [7:0]= wcheck[7:0];
 //acm_filter_RedundanceMultiplyBlock area estimate = 2195.424 representing 29.087706

  //acm_filter_MultiplyBlock area estimate = 7547.60147888824;
endmodule //acm_filter_MultiplyBlock
```

```
71
72  module acm_filter (
73      inData,
74      clk,
75      outData,
76      reset
77  );
78
79    // Port mode declarations:
80    input   [31:0] inData;
81    input    clk;
82    output  [31:0] outData;
83    input    reset;
84
85    //registerIn
86    reg [31:0] inData_in;
87
88    always@(posedge clk or negedge reset) begin
89      if(~reset) begin
90        inData_in <= 32'h00000000;
91      end  else begin
92        inData_in <= inData;
93      end
94    end
95
96    //registerOut
97    reg [31:0] outData;
98    wire [31:0] outData_in;
99
100   always@(posedge clk or negedge reset) begin
101     if(~reset) begin
102       outData <= 32'h00000000;
103     end  else begin
104       outData <= outData_in;
105     end
106   end
107
108   wire [31:0] multProducts [0:2];
109
110 integer file;
111   acm_filter_MultiplyBlock my_acm_filter_MultiplyBlock(
112 .check(check_out),
113     .X(inData_in),
114     .Y1(multProducts[0]),
115     .Y2(multProducts[1]),
116     .Y3(multProducts[2])
117   );
118
119   reg [31:0] firStep[0:1];
120
121   always@(posedge clk or negedge reset) begin
122     if(~reset) begin
123       firStep[0] <= 32'h00000000;
124       firStep[1] <= 32'h00000000;
125     end
126     else begin
127 file = $fopen("C:/Modeltech_pe_edu_10.1b/test_output.dat");
128  $fwrite(file,"%x\n",check_out);
129       firStep[0] <=  multProducts[0]; // 2448.23046908235 = flop(0, 31)
130       firStep[1] <=  firStep[0] + multProducts[1]; // 4643.65452699117 = flop(0, 31) + adder(0, 31)
131     end
132   end
133
134   assign outData_in = firStep[1]+ multProducts[2]; // 2195.42405790882 = adder(0, 31)
135   //acm_filter area estimate = 21731.3714710353;
136 endmodule //acm_filter
137
138
```

Anexo A-5 Código DOT presentado por DetectError

```
________Lectura de datos Filtro FIR
assign w16  =w1   <<    4;    //shl(4,35)
assign w17  =w1   +     w16;  //2195.42405790882
assign w221 =w255 -     w34;  //2943.86389821912
assign w255 =w256 -     w1;   //2408.3135227603
assign w256 =w1   <<    8;    //shl(8,39)
assign w34  =w17  <<    1;    //shl(1,37)
salida:1
salida:2
salida:3
```

_____Grafo bloque multiplicador Spiral

```
digraph "CFG for MCM Spiral"{ label="CFG for MCM Spiral"
Node0x1[shape=record,label="{1}"];
Node0x16[shape=record,label="{16|{<s0>4sl(16)}}"];
Node0x16:s0 -> Node0x1;
Node0x17[shape=record,label="{17|{<s0>+|<s1>+}}"];
Node0x17:s0 -> Node0x1;
Node0x17:s1 -> Node0x16;
Node0x221[shape=record,label="{221|{<s0>+|<s1>-}}"];
Node0x221:s0 -> Node0x255;
Node0x221:s1 -> Node0x34;
Node0x255[shape=record,label="{255|{<s0>+|<s1>-}}"];
Node0x255:s0 -> Node0x256;
Node0x255:s1 -> Node0x1;
Node0x256[shape=record,label="{256|{<s0>8sl(256)}}"];
Node0x256:s0 -> Node0x1;
Node0x34[shape=record,label="{34|{<s0>1sl(2)}}"];
Node0x34:s0 -> Node0x17;
Node0xY0[shape=record,label="{Y0|{<s0>+}}"];
Node0xY0:s0 -> Node0x17;
Node0xY1[shape=record,label="{Y1|{<s0>+}}"];
Node0xY1:s0 -> Node0x221;
Node0xY2[shape=record,label="{Y2|{<s0>+}}"];
Node0xY2:s0 -> Node0x17;
}
```

_____Análisis del comportamiento de nodos frente a compensación de error

```
*)salida:-3    signo:+    por:17
*)salida:-2    signo:+    por:221
*)salida:-1    signo:+    por:17

17_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
17    34    -2    -1    -1    -34
17    -3    -3    0     1     17
17    -1    -1    0     1     17
___17_afecta:0
!!!!!!!!!!!!!!!!!!!!!!!clonando:17
cambiar succ: 17b<-1
cambiar succ: 17b<-16
algo cambia17     1?=17
```

```
221_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
221   -2    -2    0     1     221
___221_afecta:221

100017_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
100017      34    -2    -1    -1    -34
___100017_afecta:-34

16_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
16    17    -1    0     1     16
16    100017      -2    -1    -1    -32
16    17    -3    0     1     16
___16_afecta:0
!!!!!!!!!!!!!!!!!!!!!!clonando:16
cambiar succ: 16b<-1
algo cambia1      16?=16

255_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
255   221   -2    0     1     255
___255_afecta:255

34_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
34    221   -2    0     -1    -34
___34_afecta:-34

256_____________________________________________
name  succ  -exit-1     s*desp      sign  afecta
256   255   -2    0     1     256
___256_afecta:256

________Área utilizada por los nodos
1     null
16    0.0
17    2195.424
221   2943.864
255   2408.3135
256   0.0
34    0.0
Y[0]  null
Y[1]  null
Y[2]  null
17b   2195.424
16b   0.0
```

_____Grafo bloque multiplicador sin compensación de error

```
digraph "CFG for MCM DetectError"{  label="CFG for MCM DetectError"
Node0x1[shape=record,label="{1}"];
Node0x16[shape=record,label="{16|{<s0>4sl(16)}}"];
Node0x16:s0 -> Node0x1;
Node0x17[shape=record,label="{17|{<s0>+|<s1>+}}"];
Node0x17:s0 -> Node0x1;
Node0x17:s1 -> Node0x16;
Node0x221[shape=record,label="{221|{<s0>+|<s1>-}}"];
Node0x221:s0 -> Node0x255;
Node0x221:s1 -> Node0x34;
Node0x255[shape=record,label="{255|{<s0>+|<s1>-}}"];
Node0x255:s0 -> Node0x256;
Node0x255:s1 -> Node0x1;
Node0x256[shape=record,label="{256|{<s0>8sl(256)}}"];
Node0x256:s0 -> Node0x1;
Node0x34[shape=record,label="{34|{<s0>1sl(2)}}"];
Node0x34:s0 -> Node0x17b;
Node0xY0[shape=record,label="{Y0|{<s0>+}}"];
Node0xY0:s0 -> Node0x17;
Node0xY1[shape=record,label="{Y1|{<s0>+}}"];
Node0xY1:s0 -> Node0x221;
Node0xY2[shape=record,label="{Y2|{<s0>+}}"];
Node0xY2:s0 -> Node0x17;
Node0x17b[shape=record,label="{17b|{<s0>+|<s1>+}}"];
Node0x17b:s0 -> Node0x1;
Node0x17b:s1 -> Node0x16b;
Node0x16b[shape=record,label="{16b|{<s0>4sl(16)}}"];
Node0x16b:s0 -> Node0x1;
}
```

_____Estudio area utilizada

```
inicial     7547.6016
overhead    2195.424
overhead%   29.087706%
final       9743.025
```

_____

_!!_____

_____Script ModelSim

```
force w1 0;run 50; noforce w1;run 50;force w16 0;run 50; noforce w16;run
50;force w17 0;run 50; noforce w17;run 50;force w221 0;run 50; noforce w221;run
50;force w255 0;run 50; noforce w255;run 50;force w256 0;run 50; noforce
w256;run 50;force w34 0;run 50; noforce w34;run 50;force w17b 0;run 50; noforce
w17b;run 50;force w16b 0;run 50; noforce w16b;run 50;
```

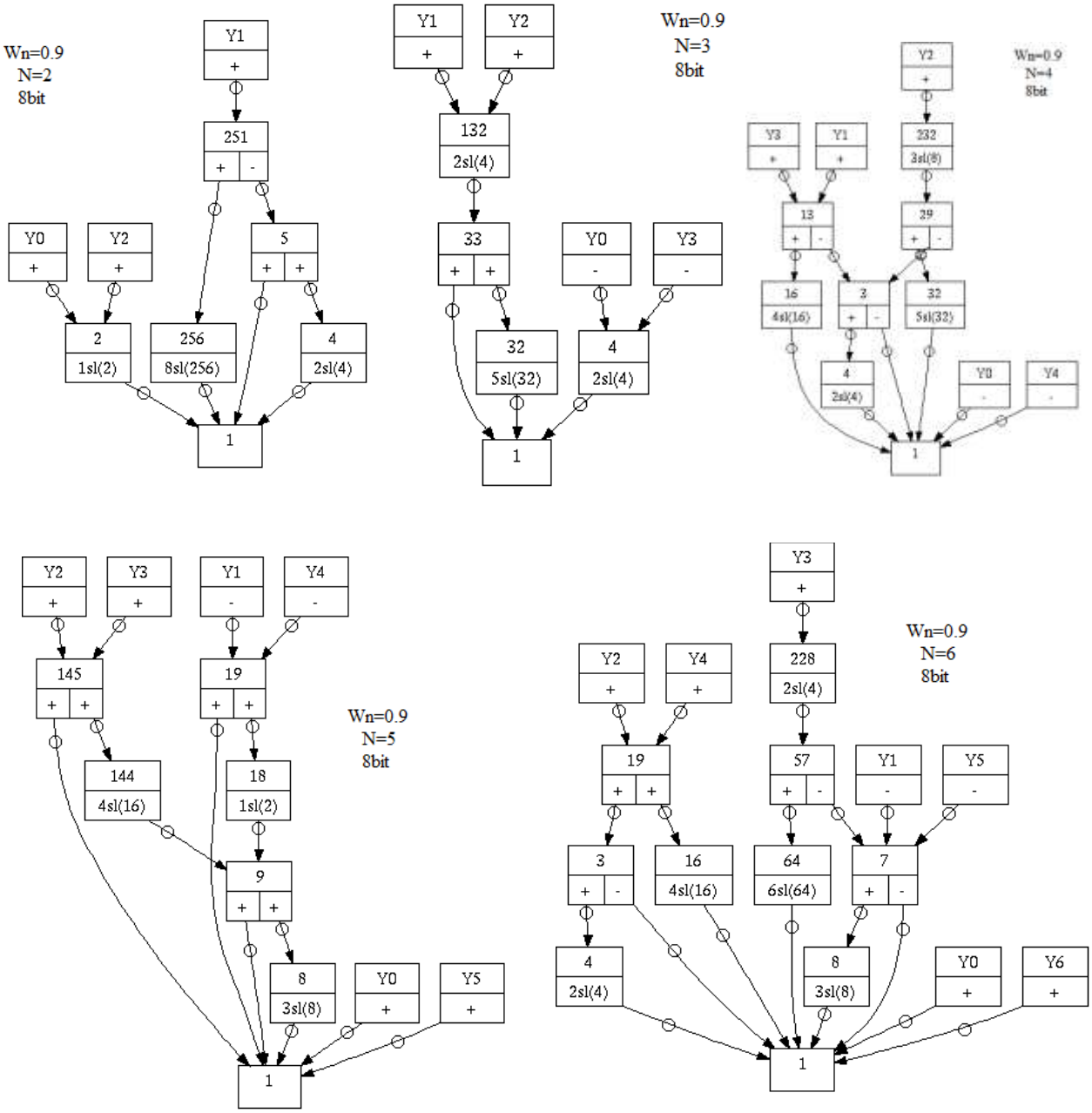
_____InfoSize

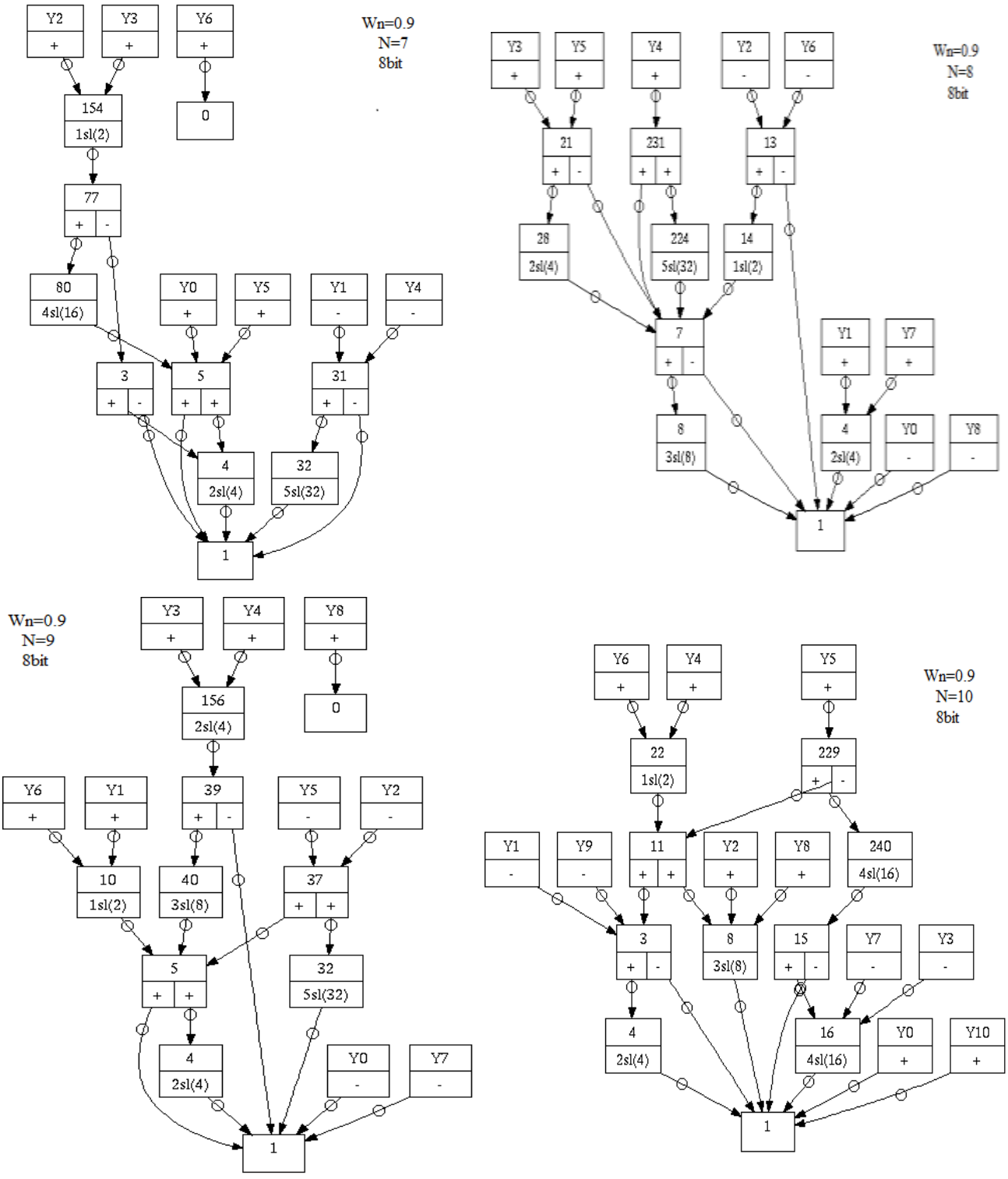
```
  //acm_filter_MultiplyBlock area estimate = 7547.60147888824;
  //acm_filter area estimate = 21731.3714710353;
endmodule //acm_filter
Done.
```

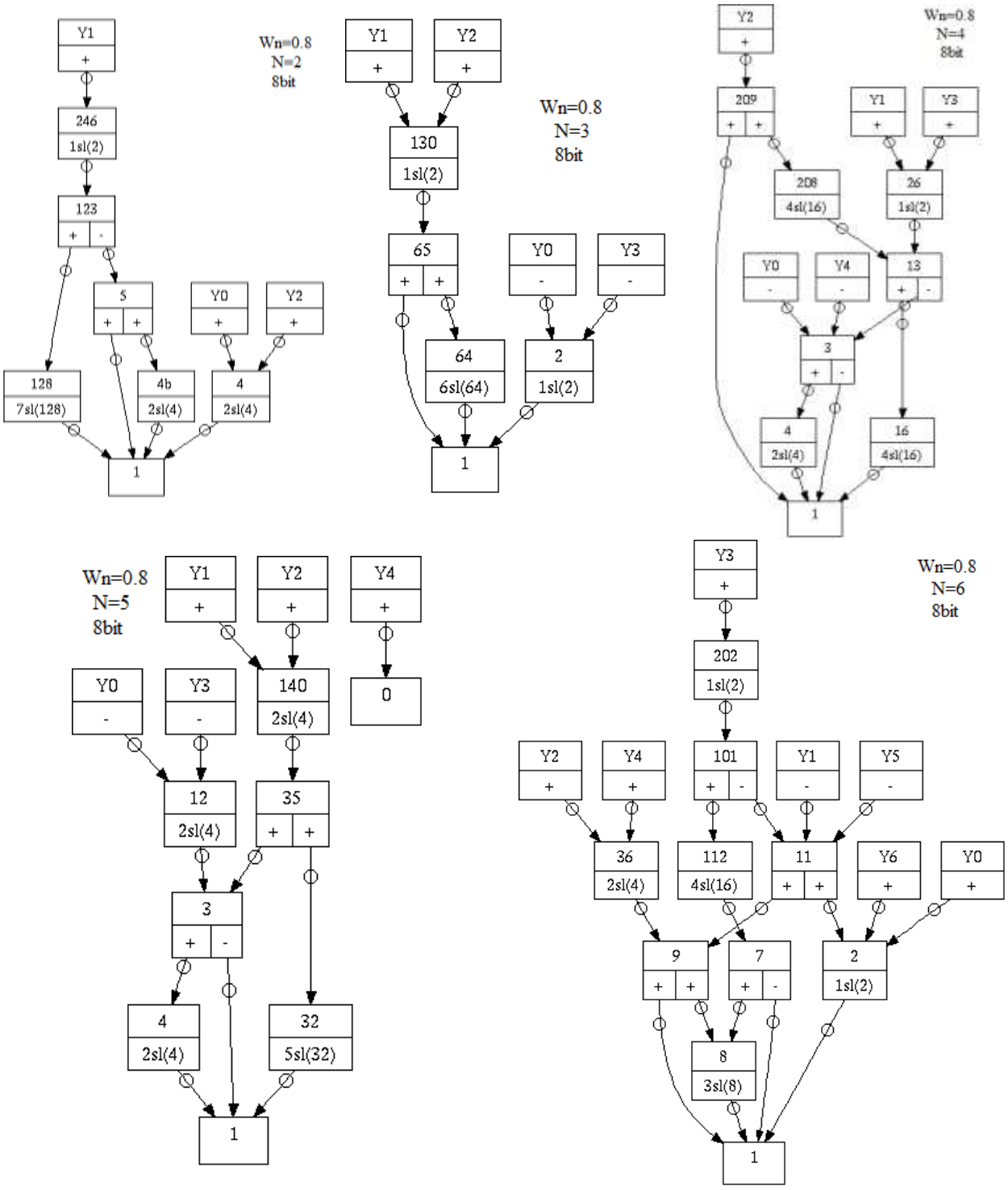
Anexo A-6 Grafos de Filtros FIR

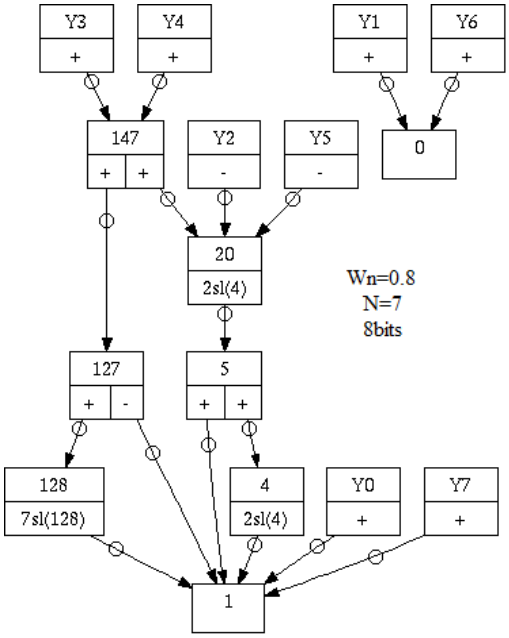
En este Anexo se recogen los grafos calculados con la aplicación dotty, correspondientes a los Bloques Multiplicadores de los Filtros FIR Paso bajo descritos en este Proyecto. En cada caso se indica la Frecuencia de corte y el orden del filtro.

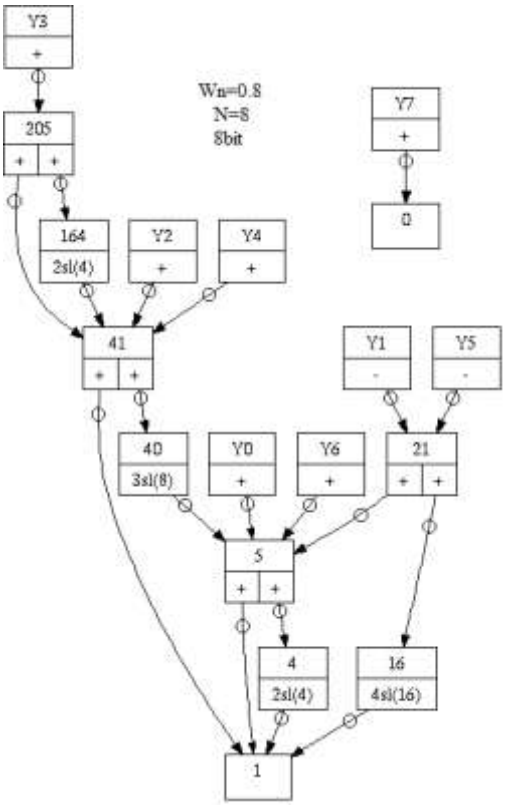
A-6.1 Grafos de Filtros FIR Spiral Paso bajo 8 bit de precisión.

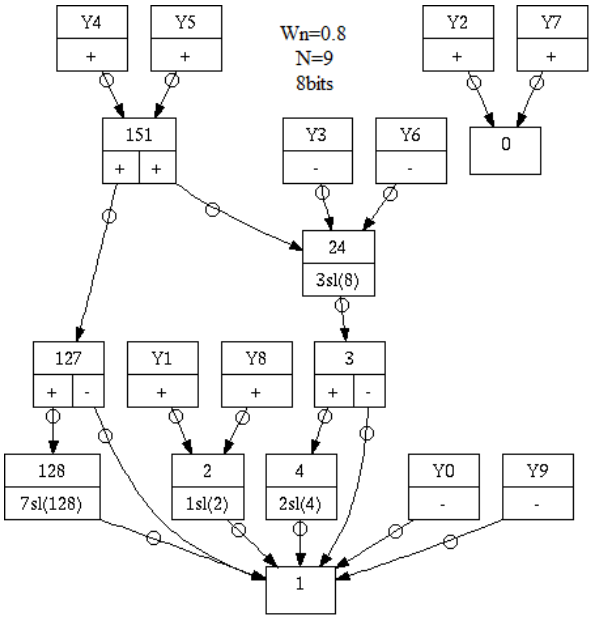


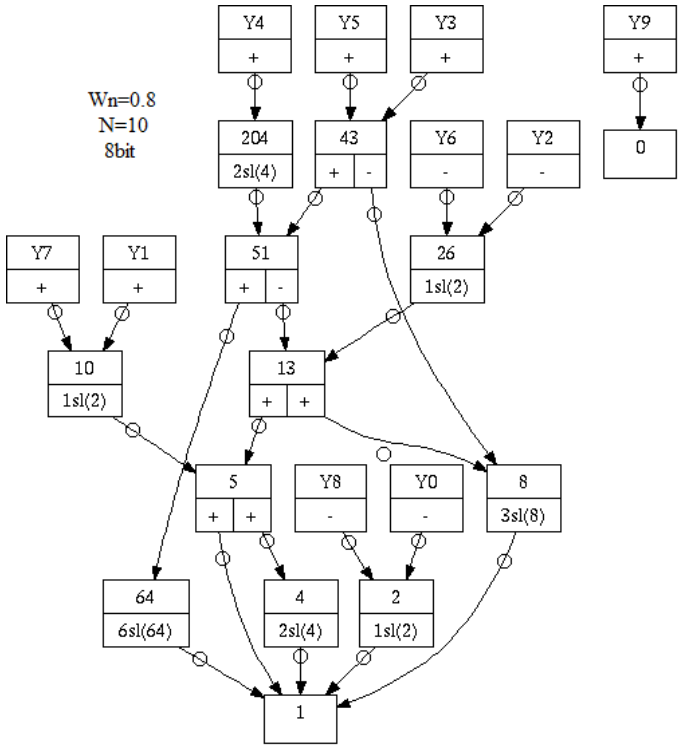

Wn=0.9
N=7
8bit
Y2
Y3
Y6
154
1sl(2)
0
77
80
4sl(16)
Y0
Y5
Y1
Y4
3
5
31
4
2sl(4)
32
5sl(32)
1
Wn=0.9
N=8
8bit
Y3
Y5
Y4
Y2
Y6
21
231
13
28
2sl(4)
224
5sl(32)
14
1sl(2)
7
Y1
Y7
8
3sl(8)
4
2sl(4)
Y0
Y8
1
Wn=0.9
N=9
8bit
Y3
Y4
Y8
156
2sl(4)
0
Y6
Y1
39
Y5
Y2
10
1sl(2)
40
3sl(8)
37
5
32
5sl(32)
4
2sl(4)
Y0
Y7
1
Wn=0.9
N=10
8bit
Y6
Y4
Y5
22
1sl(2)
229
Y1
Y9
11
Y2
Y8
240
4sl(16)
3
8
3sl(8)
15
Y7
Y3
4
2sl(4)
16
4sl(16)
Y0
Y10
1

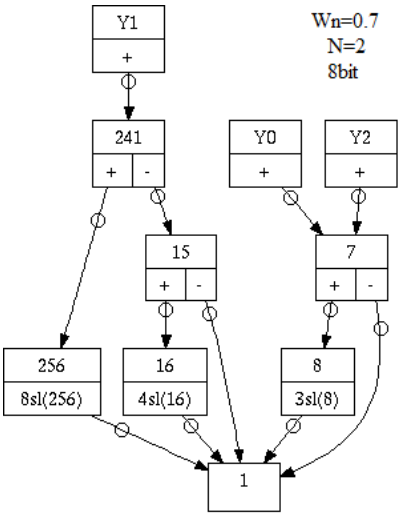

Wn=0.8
N=2
8bit
Y1
+
246
1sl(2)
123
+ -
5
+ +
Y0
+
Y2
+
128
7sl(128)
4b
2sl(4)
4
2sl(4)
1
Wn=0.8
N=3
8bit
Y1
+
Y2
+
130
1sl(2)
65
+ +
Y0
-
Y3
-
64
6sl(64)
2
1sl(2)
1
Wn=0.8
N=4
8bit
Y2
+
209
+ +
Y1
+
Y3
+
208
4sl(16)
26
1sl(2)
Y0
-
Y4
-
13
+ -
3
+ -
4
2sl(4)
16
4sl(16)
1
Wn=0.8
N=5
8bit
Y1
+
Y2
+
Y4
+
Y0
-
Y3
-
140
2sl(4)
0
12
2sl(4)
35
+ +
3
+ -
4
2sl(4)
32
5sl(32)
1
Wn=0.8
N=6
8bit
Y3
+
202
1sl(2)
Y2
+
Y4
+
101
+ -
Y1
-
Y5
-
36
2sl(4)
112
4sl(16)
11
+ +
Y6
+
Y0
+
9
+ +
7
+ -
2
1sl(2)
8
3sl(8)
1


Y3
Y4
Y1
Y6
147
Y2
Y5
0
20
2sl(4)
Wn=0.8
N=7
8bits
127
5
128
7sl(128)
4
2sl(4)
Y0
Y7
1

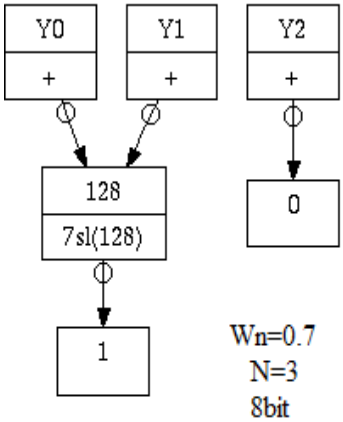

Y3
Wn=0.8
N=8
8bit
205
Y7
0
164
2sl(4)
Y2
Y4
41
Y1
Y5
40
3sl(8)
Y0
Y6
21
5
4
2sl(4)
16
4sl(16)
1


Y4
Y5
Wn=0.8
N=9
8bits
Y2
Y7
151
Y3
Y6
0
24
3sl(8)
127
Y1
Y8
3
128
7sl(128)
2
1sl(2)
4
2sl(4)
Y0
Y9
1

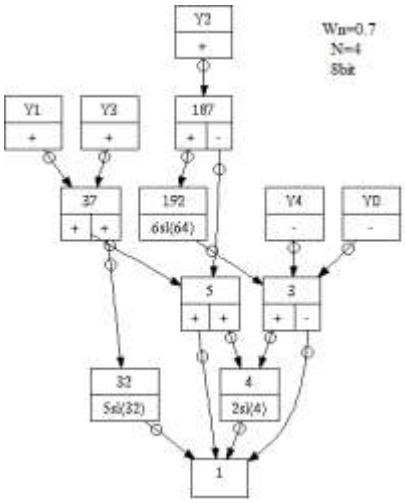

Y4
Y5
Y3
Y9
Wn=0.8
N=10
8bit
204
2sl(4)
43
Y6
Y2
0
Y7
Y1
51
26
1sl(2)
10
1sl(2)
13
5
Y8
Y0
8
3sl(8)
64
6sl(64)
4
2sl(4)
2
1sl(2)
1

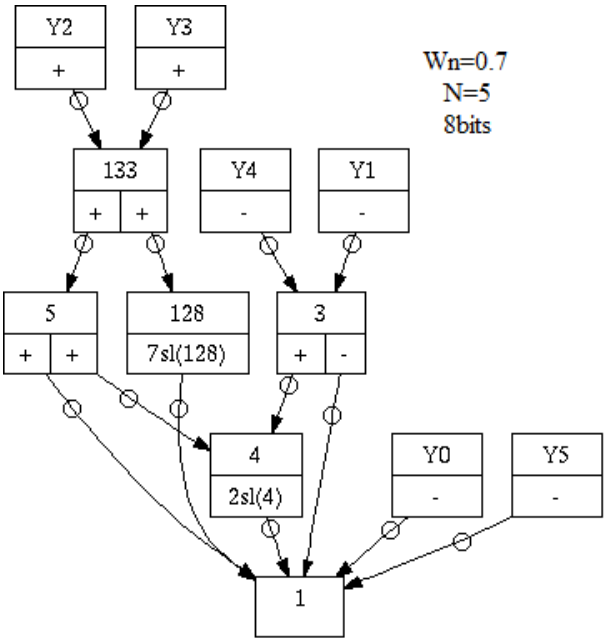


Wn=0.7
N=2
8bit

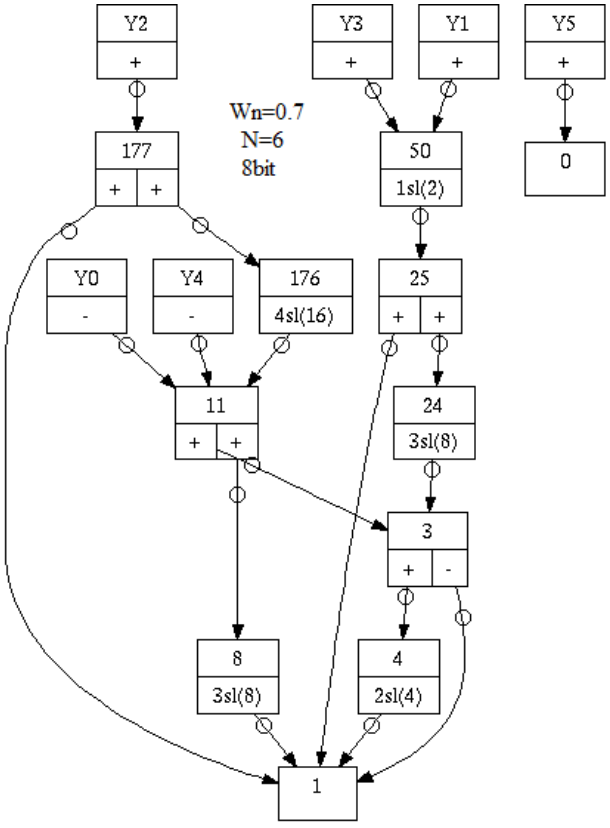


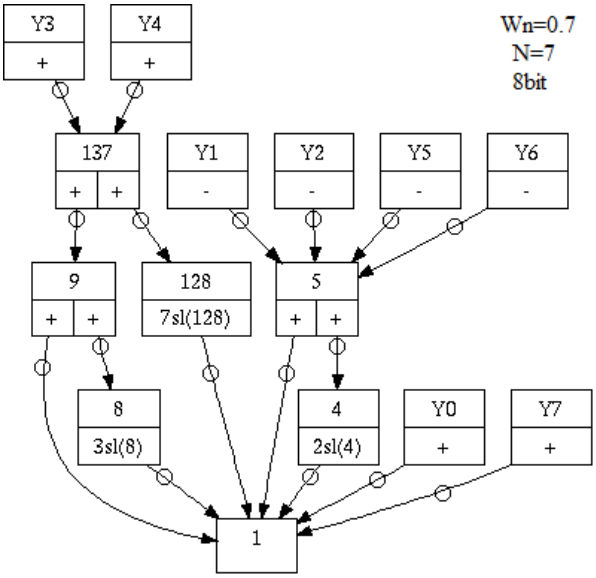
Wn=0.7
N=3
8bit

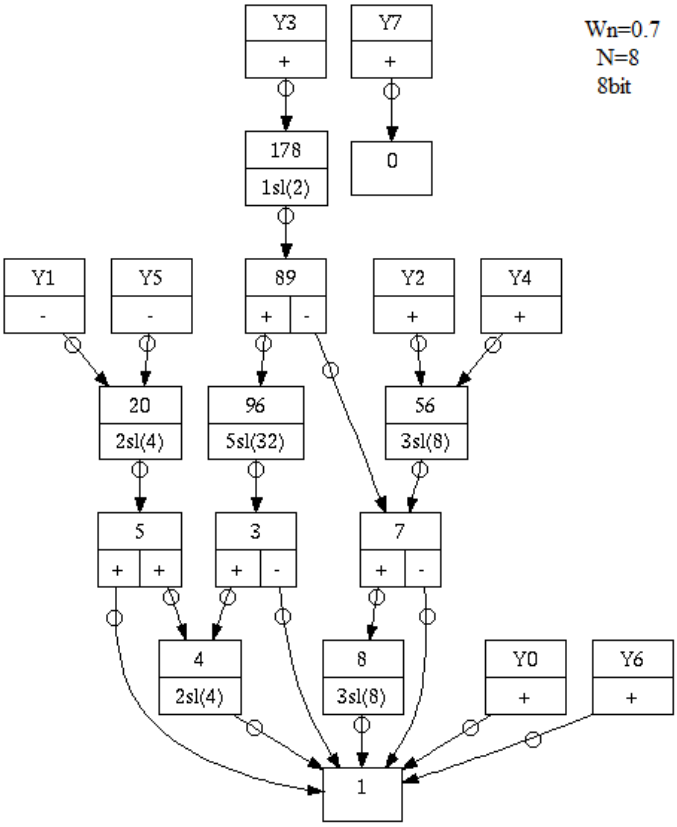


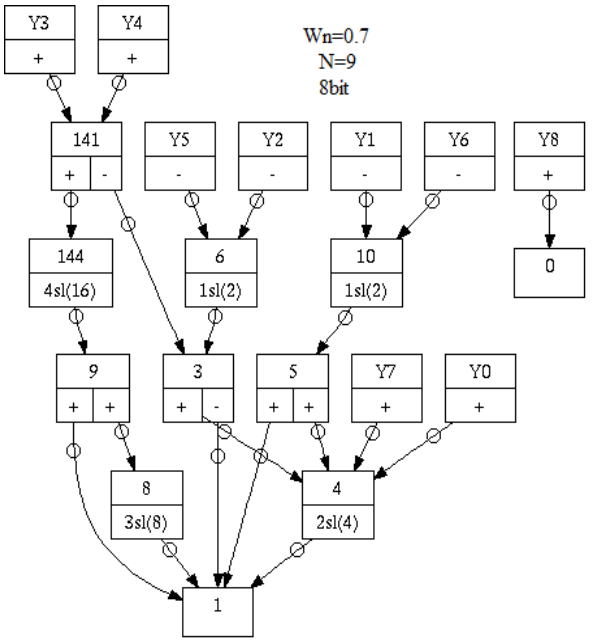


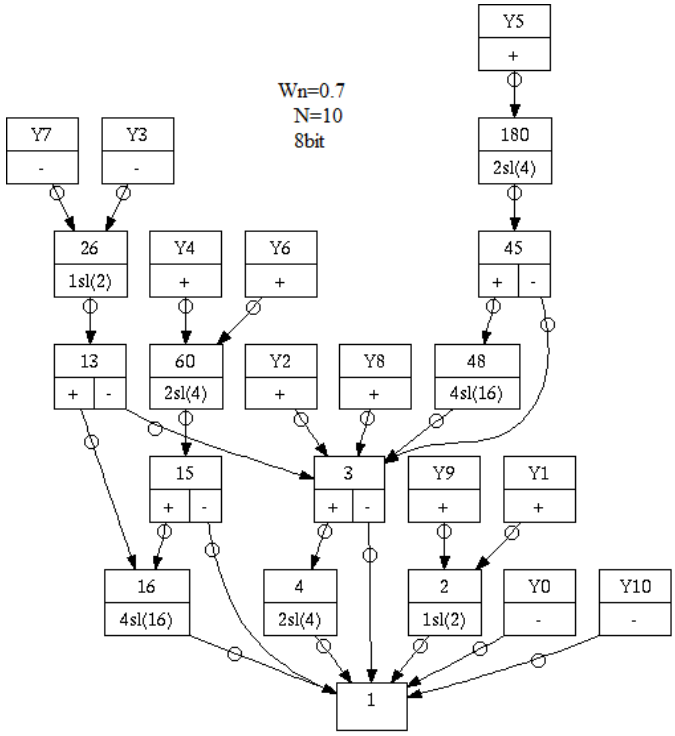
Wn=0.7
N=5
8bits

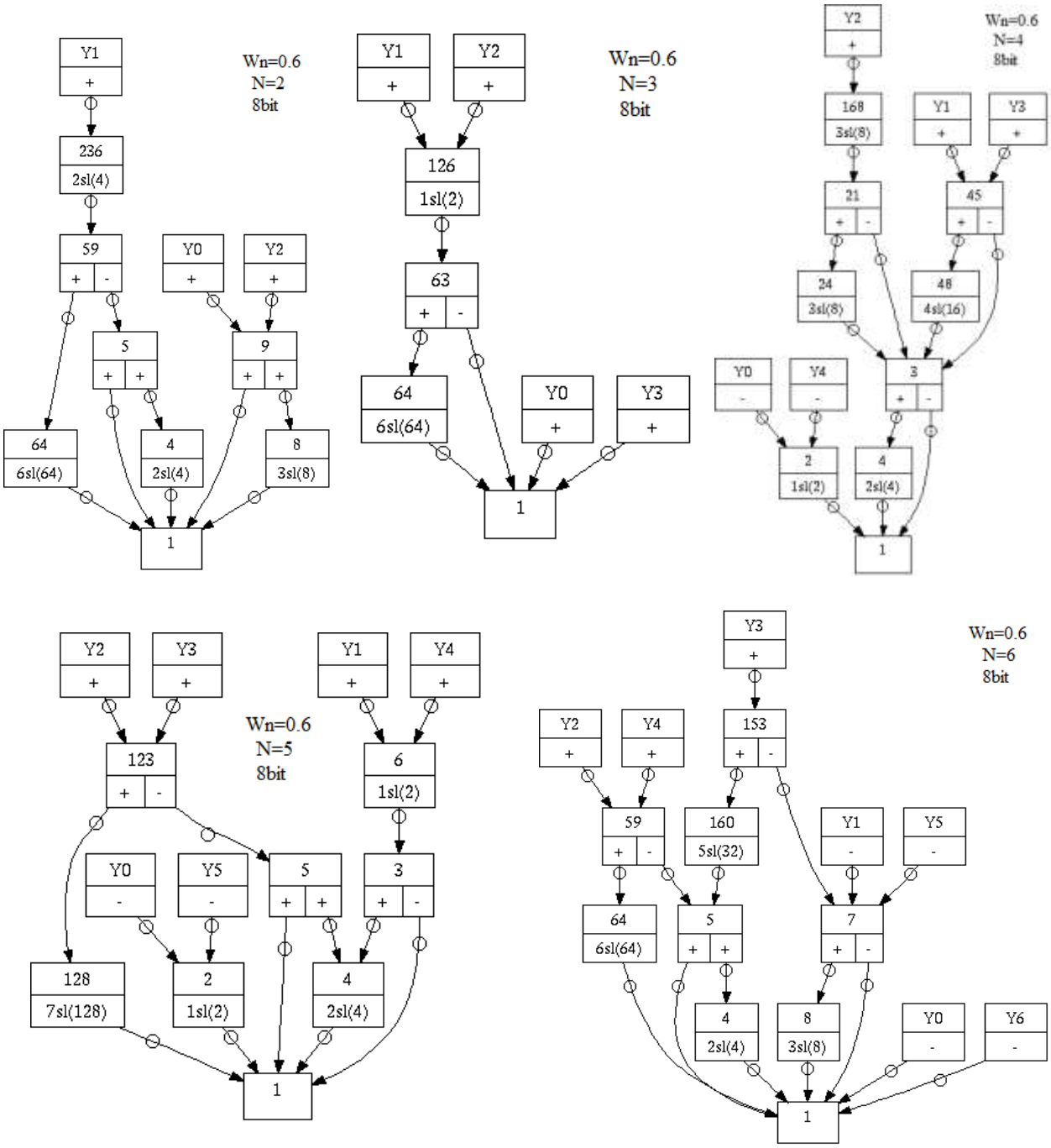


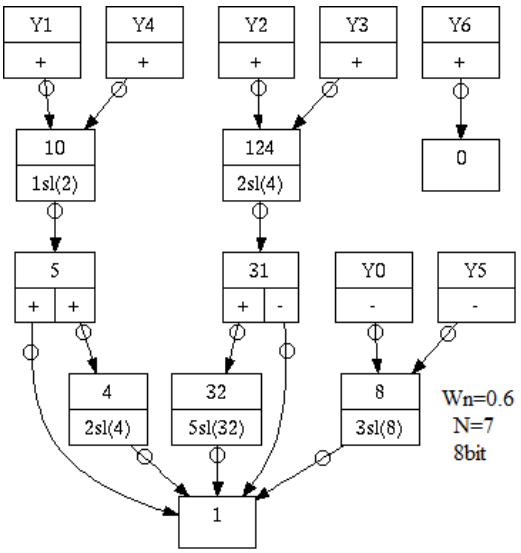

Wn=0.7
N=7
8bit
Y3
Y4
137
Y1
Y2
Y5
Y6
9
128
7sl(128)
5
8
3sl(8)
4
2sl(4)
Y0
Y7
1

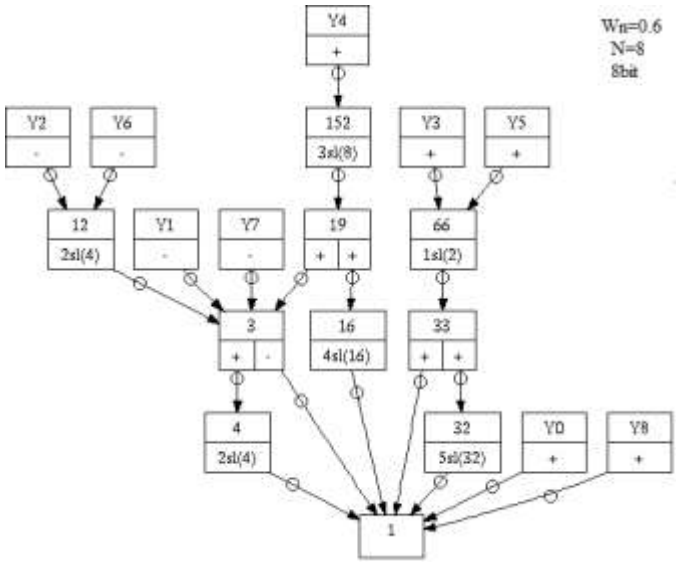

Wn=0.7
N=8
8bit
Y3
Y7
178
1sl(2)
0
Y1
Y5
89
Y2
Y4
20
2sl(4)
96
5sl(32)
56
3sl(8)
5
3
7
4
2sl(4)
8
3sl(8)
Y0
Y6
1

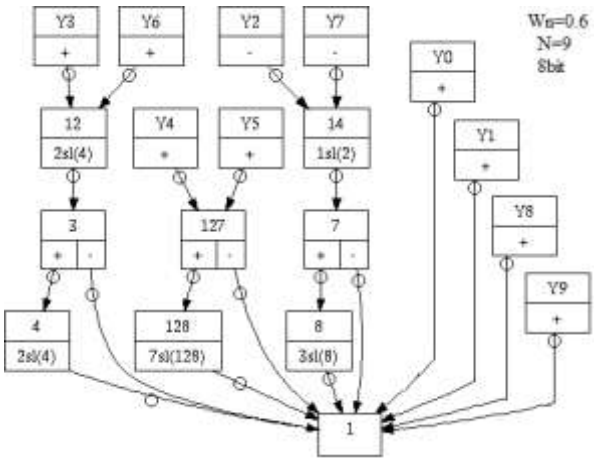

Wn=0.7
N=9
8bit
Y3
Y4
141
Y5
Y2
Y1
Y6
Y8
144
4sl(16)
6
1sl(2)
10
1sl(2)
0
9
3
5
Y7
Y0
8
3sl(8)
4
2sl(4)
1

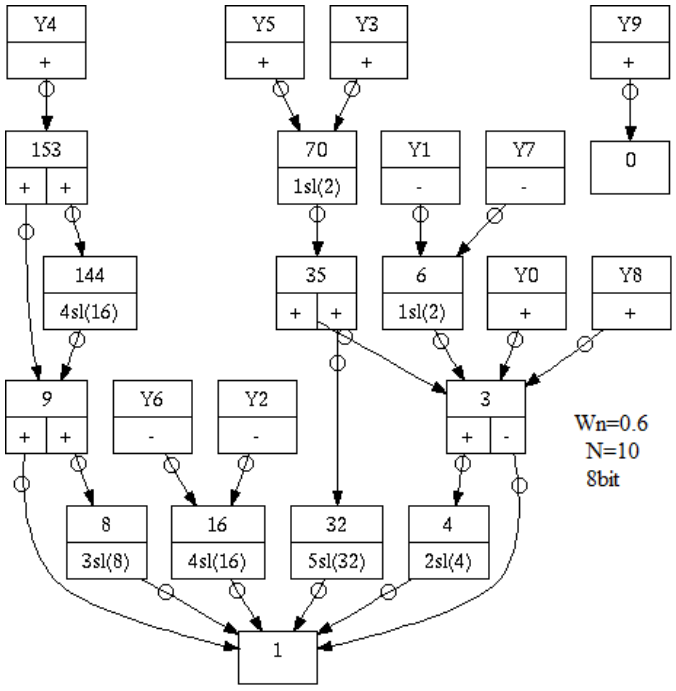

Wn=0.7
N=10
8bit
Y5
Y7
Y3
180
2sl(4)
26
1sl(2)
Y4
Y6
45
13
60
2sl(4)
Y2
Y8
48
4sl(16)
15
3
Y9
Y1
16
4sl(16)
4
2sl(4)
2
1sl(2)
Y0
Y10
1

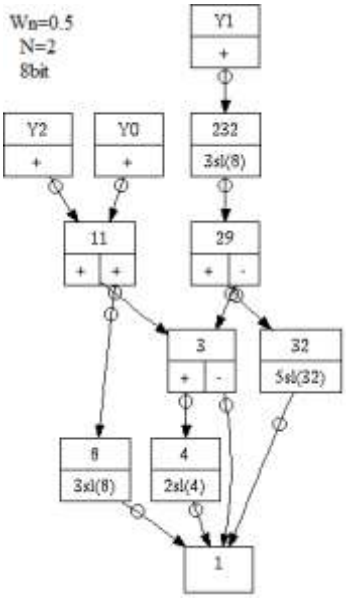

Wn=0.6
N=2
8bit
Y1
+
236
2sl(4)
59
+ -
Y0
+
Y2
+
5
+ +
9
+ +
64
6sl(64)
4
2sl(4)
8
3sl(8)
1
Wn=0.6
N=3
8bit
Y1
+
Y2
+
126
1sl(2)
63
+ -
64
6sl(64)
Y0
+
Y3
+
1
Wn=0.6
N=4
8bit
Y2
+
168
3sl(8)
Y1
+
Y3
+
21
+ -
45
+ -
24
3sl(8)
48
4sl(16)
Y0
-
Y4
-
3
+ -
2
1sl(2)
4
2sl(4)
1
Wn=0.6
N=5
8bit
Y2
+
Y3
+
Y1
+
Y4
+
123
+ -
6
1sl(2)
Y0
-
Y5
-
5
+ +
3
+ -
128
7sl(128)
2
1sl(2)
4
2sl(4)
1
Wn=0.6
N=6
8bit
Y3
+
Y2
+
Y4
+
153
+ -
59
+ -
160
5sl(32)
Y1
-
Y5
-
64
6sl(64)
5
+ +
7
+ -
4
2sl(4)
8
3sl(8)
Y0
-
Y6
-
1

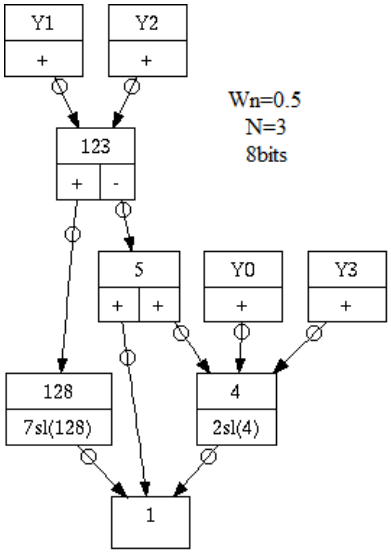

Y1
Y4
Y2
Y3
Y6
10
1sl(2)
124
2sl(4)
0
5
31
Y0
Y5
4
2sl(4)
32
5sl(32)
8
3sl(8)
1
Wn=0.6
N=7
8bit

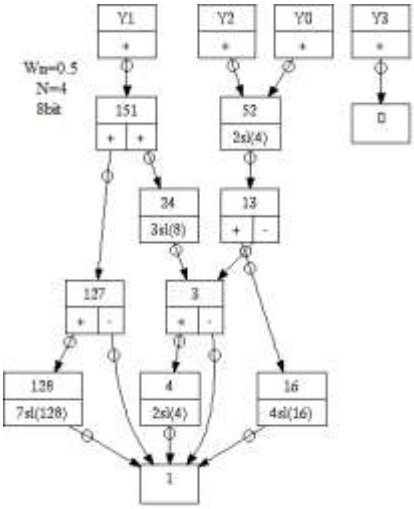

Y4
Wn=0.6
N=8
8bit
Y2
Y6
152
3sl(8)
Y3
Y5
12
2sl(4)
Y1
Y7
19
66
1sl(2)
3
16
4sl(16)
33
4
2sl(4)
32
5sl(32)
Y0
Y8
1

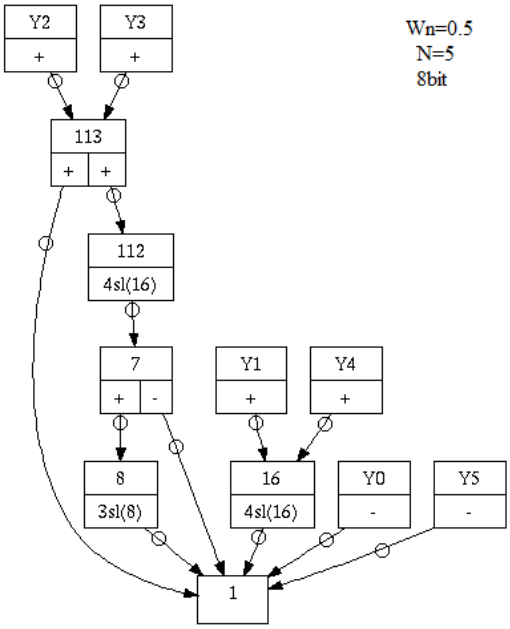

Y3
Y6
Y2
Y7
Y0
Wn=0.6
N=9
8bit
12
2sl(4)
Y4
Y5
14
1sl(2)
Y1
3
127
7
Y8
Y9
4
2sl(4)
128
7sl(128)
8
3sl(8)
1

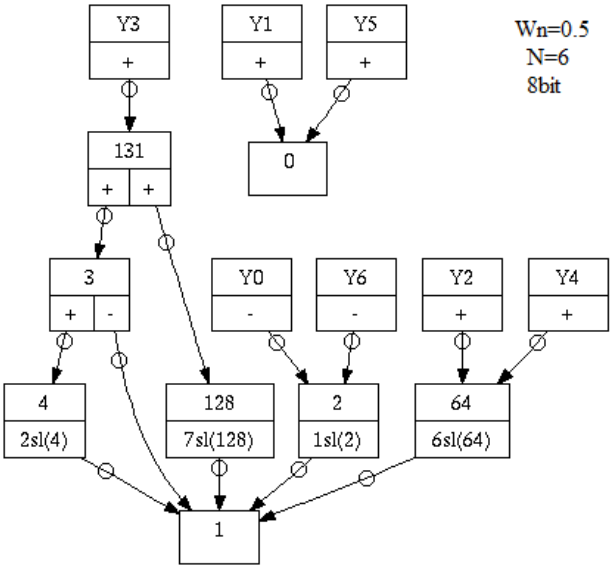

Y4
Y5
Y3
Y9
153
70
1sl(2)
Y1
Y7
0
144
4sl(16)
35
6
1sl(2)
Y0
Y8
9
Y6
Y2
3
Wn=0.6
N=10
8bit
8
3sl(8)
16
4sl(16)
32
5sl(32)
4
2sl(4)
1

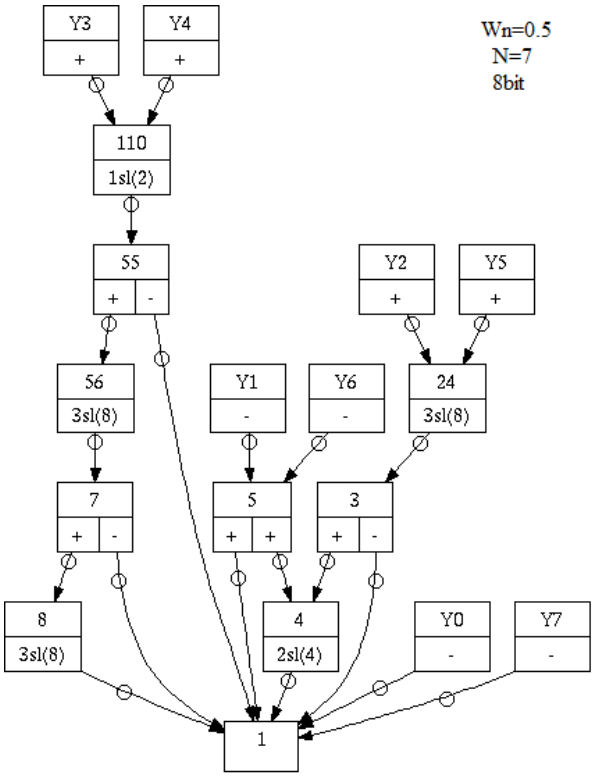

Wn=0.5
N=2
8bit
Y1
+
Y2
+
Y0
+
232
3sl(8)
11
+ +
29
+ -
3
+ -
32
5sl(32)
8
3sl(8)
4
2sl(4)
1

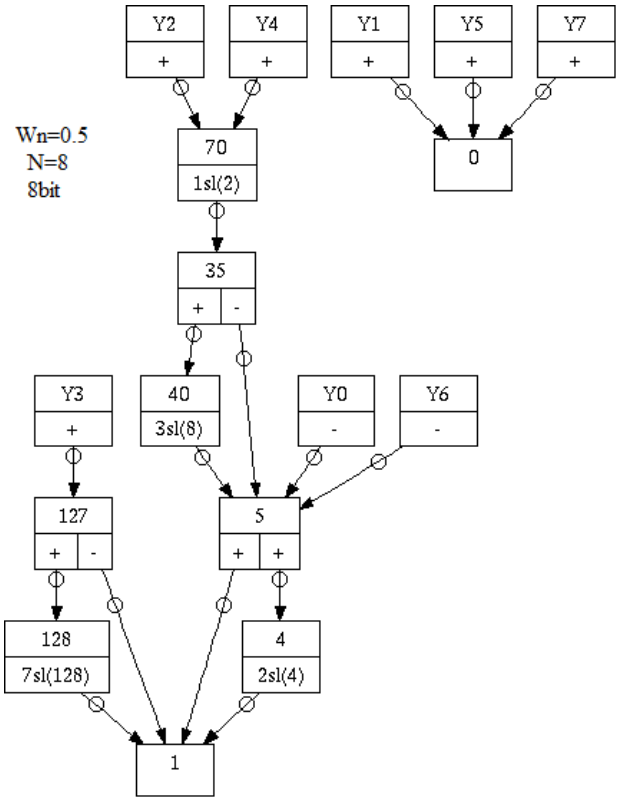

Y1
+
Y2
+
123
+ -
Wn=0.5
N=3
8bits
5
+ +
Y0
+
Y3
+
128
7sl(128)
4
2sl(4)
1

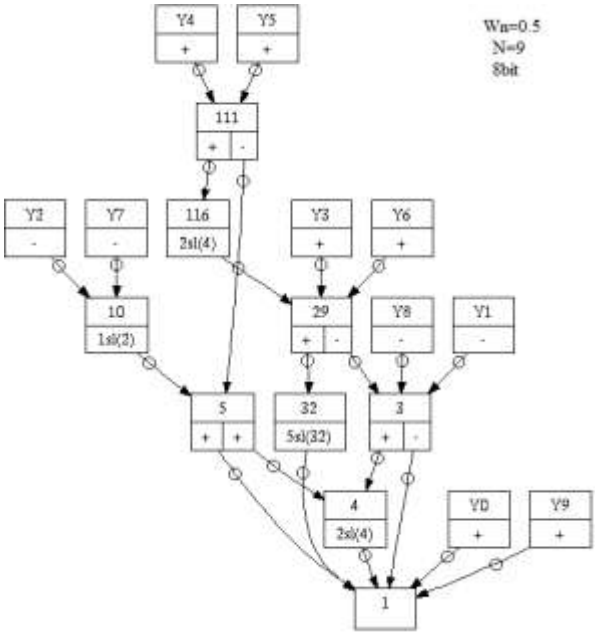

Wn=0.5
N=4
8bit
Y1
+
Y2
+
Y0
+
Y3
+
151
+ +
52
2sl(4)
0
24
3sl(8)
13
+ -
127
+ -
3
+ -
128
7sl(128)
4
2sl(4)
16
4sl(16)
1

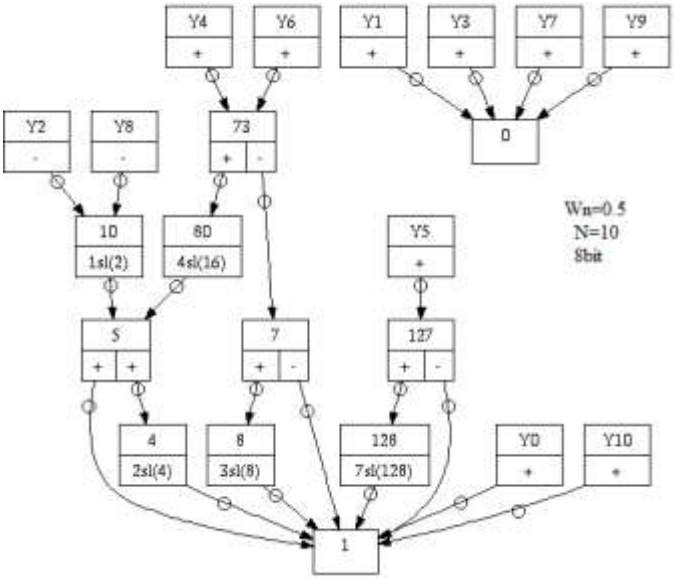

Y2
+
Y3
+
Wn=0.5
N=5
8bit
113
+ +
112
4sl(16)
7
+ -
Y1
+
Y4
+
8
3sl(8)
16
4sl(16)
Y0
-
Y5
-
1

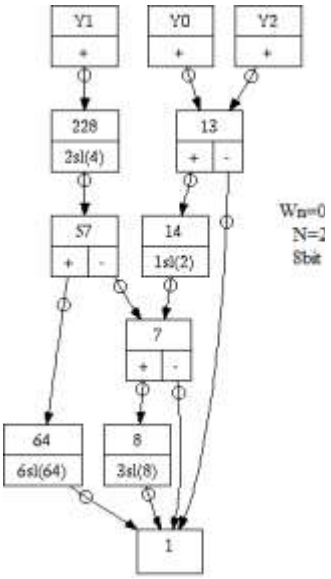

Y3
+
Y1
+
Y5
+
Wn=0.5
N=6
8bit
131
+ +
0
3
+ -
Y0
-
Y6
-
Y2
+
Y4
+
4
2sl(4)
128
7sl(128)
2
1sl(2)
64
6sl(64)
1

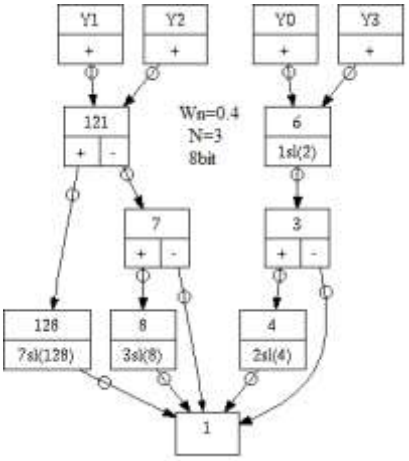

Y3
Y4
Wn=0.5
N=7
8bit
110
1sl(2)
55
Y2
Y5
56
3sl(8)
Y1
Y6
24
3sl(8)
7
5
3
8
3sl(8)
4
2sl(4)
Y0
Y7
1

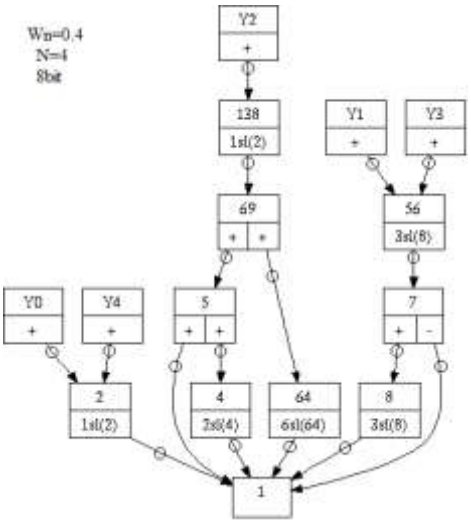

Y2
Y4
Y1
Y5
Y7
Wn=0.5
N=8
8bit
70
1sl(2)
0
35
Y3
40
3sl(8)
Y0
Y6
127
5
128
7sl(128)
4
2sl(4)
1

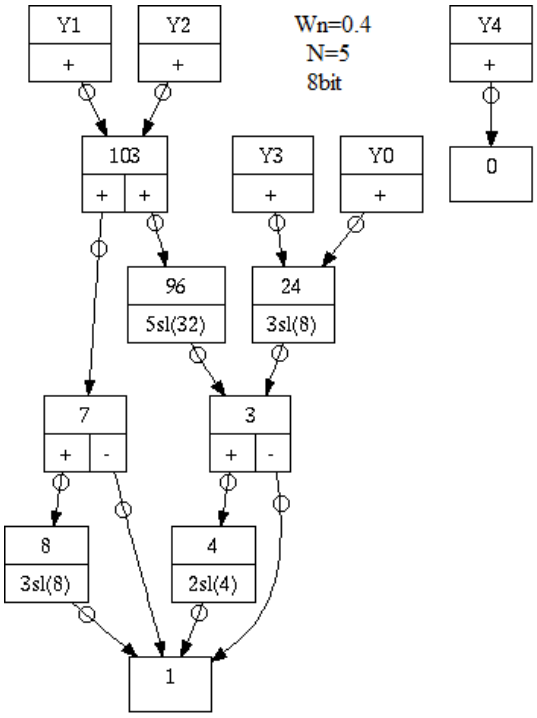

Y4
Y5
Wn=0.5
N=9
8bit
111
Y2
Y7
116
2sl(4)
Y3
Y6
10
1sl(2)
29
Y8
Y1
5
32
5sl(32)
3
4
2sl(4)
Y0
Y9
1

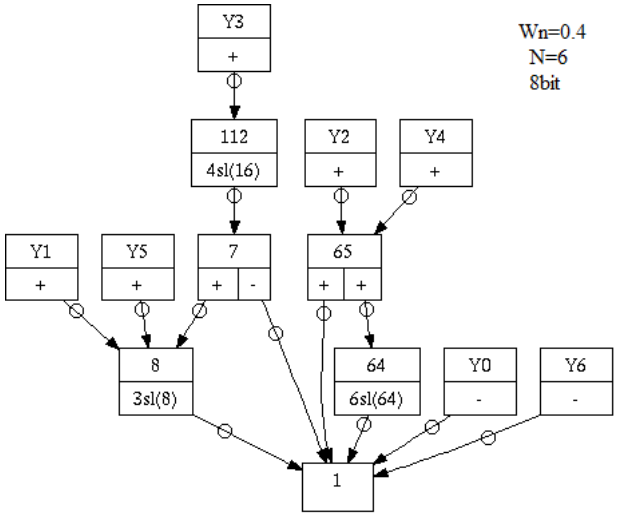

Y4
Y6
Y1
Y3
Y7
Y9
Y2
Y8
73
0
Wn=0.5
N=10
8bit
10
1sl(2)
80
4sl(16)
Y5
5
7
127
4
2sl(4)
8
3sl(8)
128
7sl(128)
Y0
Y10
1

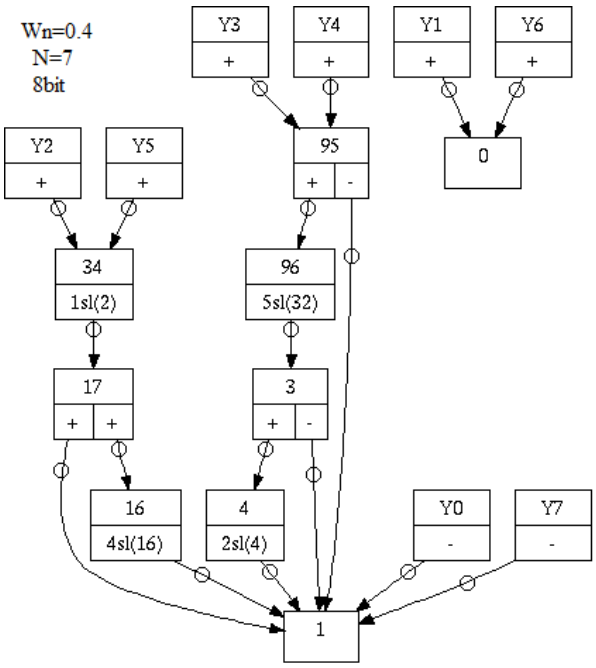

Y1
Y0
Y2
228
2sl(4)
13
57
14
1sl(2)
7
64
6sl(64)
8
3sl(8)
1

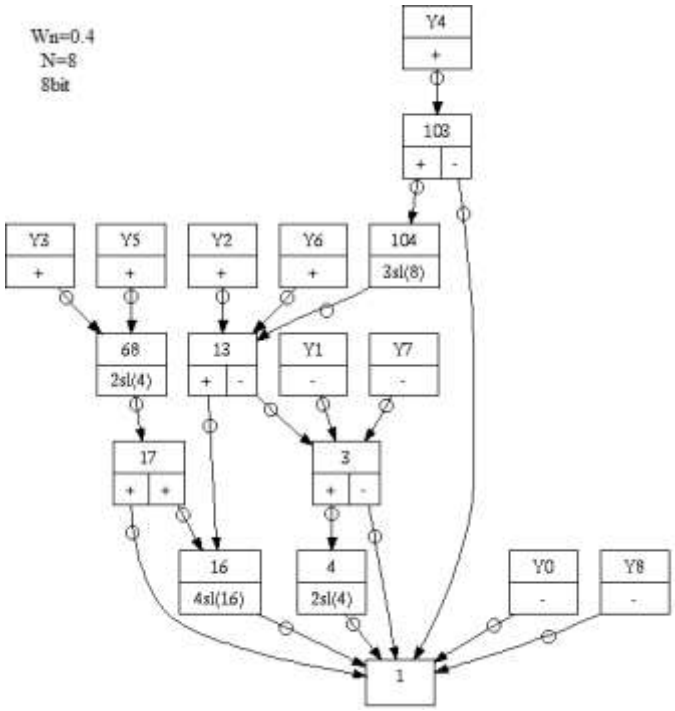

Y1
Y2
Y0
Y3
Wn=0.4
N=3
8bit
121
6
1sl(2)
7
3
126
7sl(128)
8
3sl(8)
4
2sl(4)
1

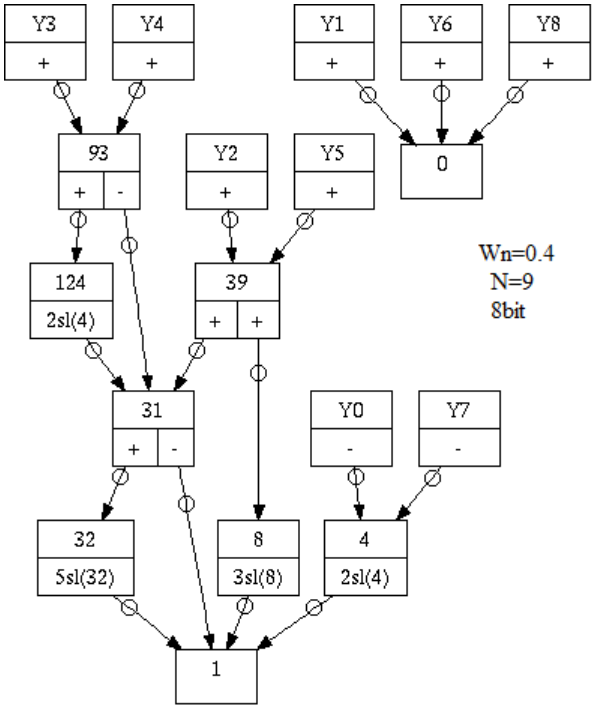

Wn=0.4
N=4
8bit
Y2
138
1sl(2)
Y1
Y3
69
56
3sl(8)
Y0
Y4
5
7
2
1sl(2)
4
2sl(4)
64
6sl(64)
8
3sl(8)
1

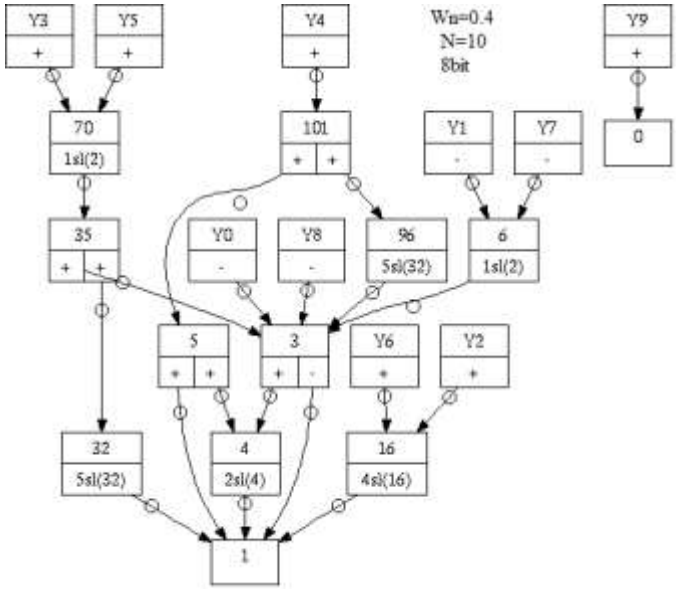

Y1
Y2
Wn=0.4
N=5
8bit
Y4
103
Y3
Y0
0
96
5sl(32)
24
3sl(8)
7
3
8
3sl(8)
4
2sl(4)
1

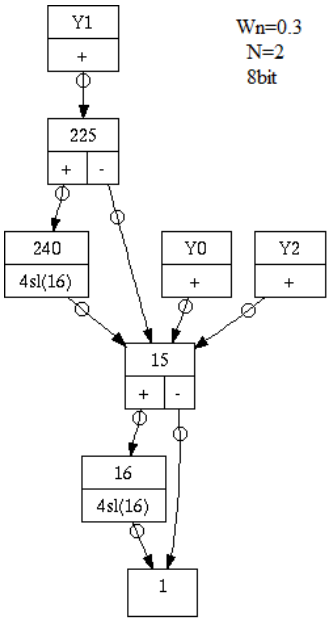

Y3
Wn=0.4
N=6
8bit
112
4sl(16)
Y2
Y4
Y1
Y5
7
65
8
3sl(8)
64
6sl(64)
Y0
-
Y6
-
1


Wn=0.4
N=7
8bit
Y3
Y4
Y1
Y6
Y2
Y5
95
0
34
1sl(2)
96
5sl(32)
17
3
16
4sl(16)
4
2sl(4)
Y0
Y7
1

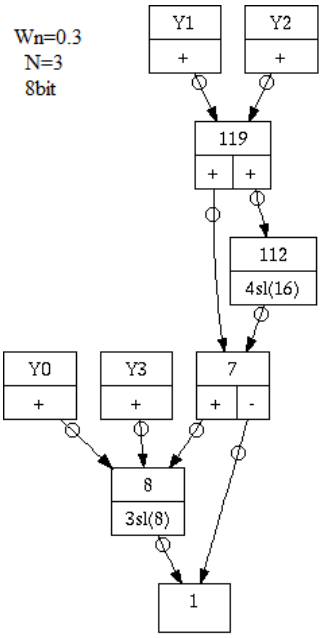

Wn=0.4
N=8
8bit
Y4
103
Y3
Y5
Y2
Y6
104
3sl(8)
68
2sl(4)
13
Y1
Y7
17
3
16
4sl(16)
4
2sl(4)
Y0
Y8
1


Y3
Y4
Y1
Y6
Y8
93
Y2
Y5
0
Wn=0.4
N=9
8bit
124
2sl(4)
39
31
Y0
Y7
32
5sl(32)
8
3sl(8)
4
2sl(4)
1

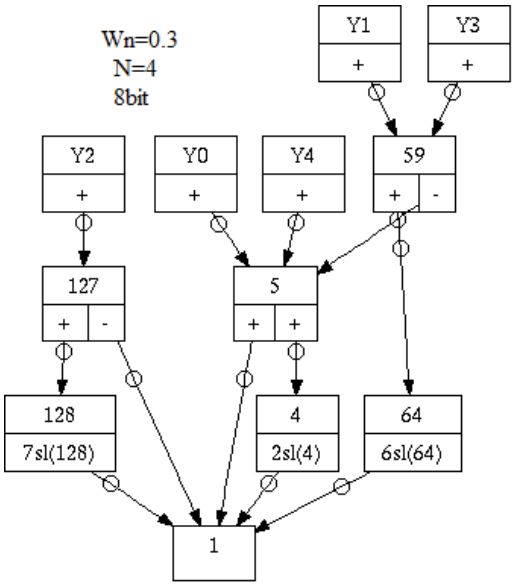

Y3
Y5
Y4
Wn=0.4
N=10
8bit
Y9
70
1sl(2)
101
Y1
Y7
0
35
Y0
Y8
96
5sl(32)
6
1sl(2)
5
3
Y6
Y2
32
5sl(32)
4
2sl(4)
16
4sl(16)
1



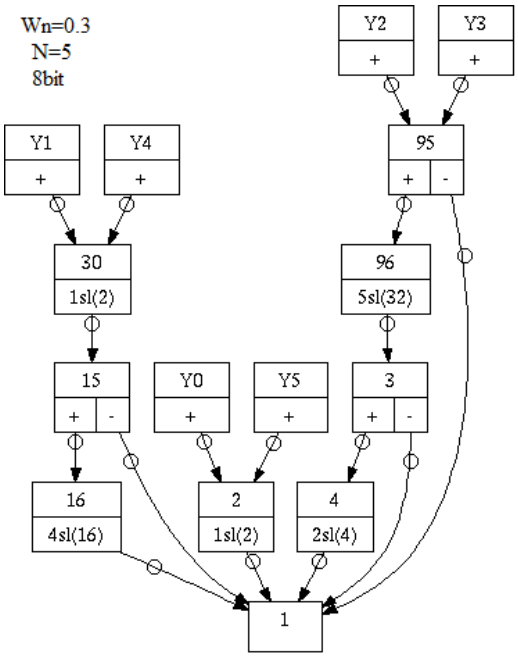
Wn=0.3
N=2
8bit



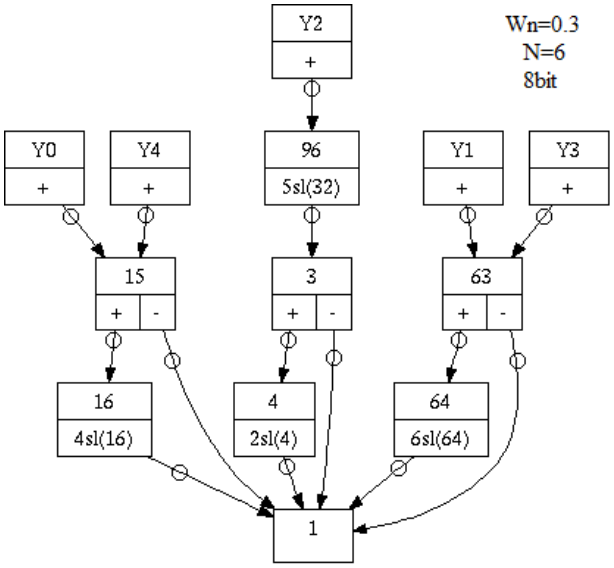
Wn=0.3
N=3
8bit



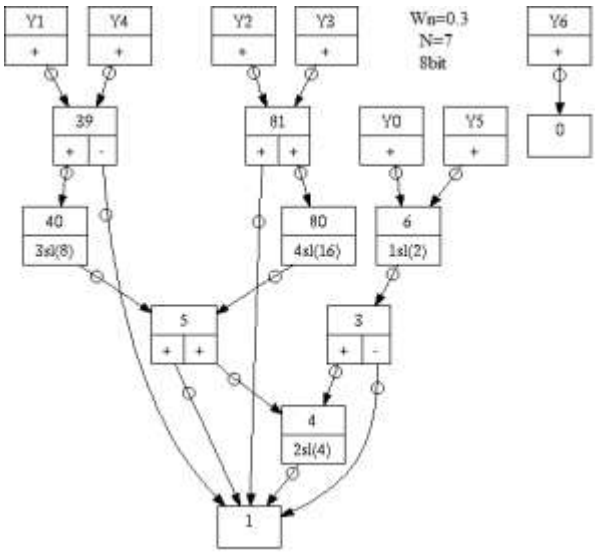
Wn=0.3
N=4
8bit

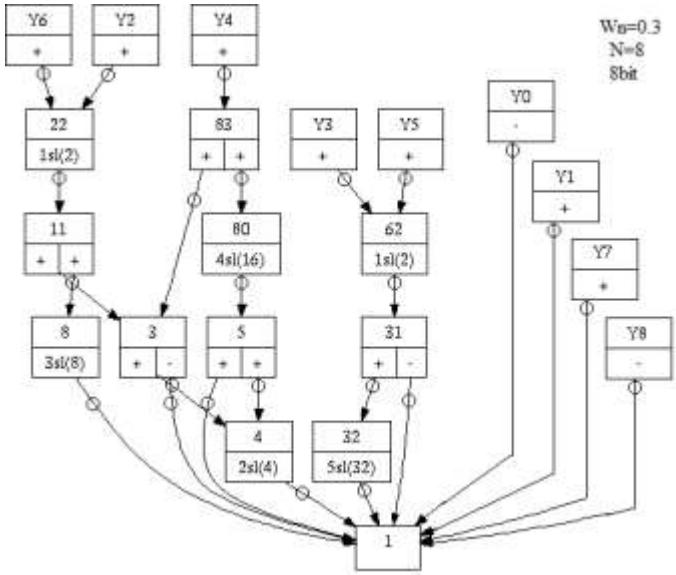


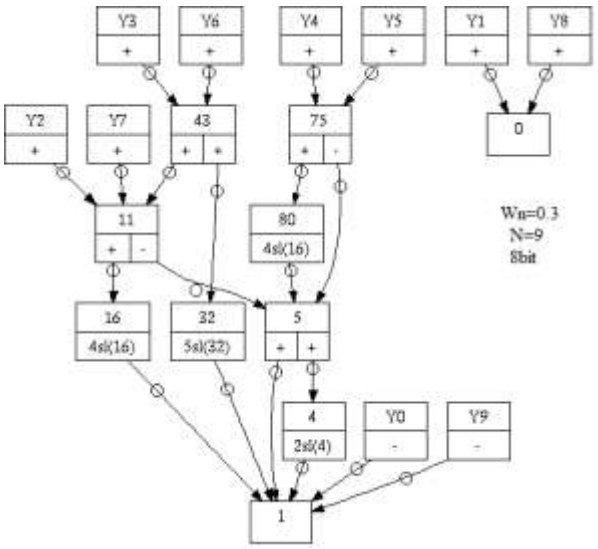
Wn=0.3
N=5
8bit

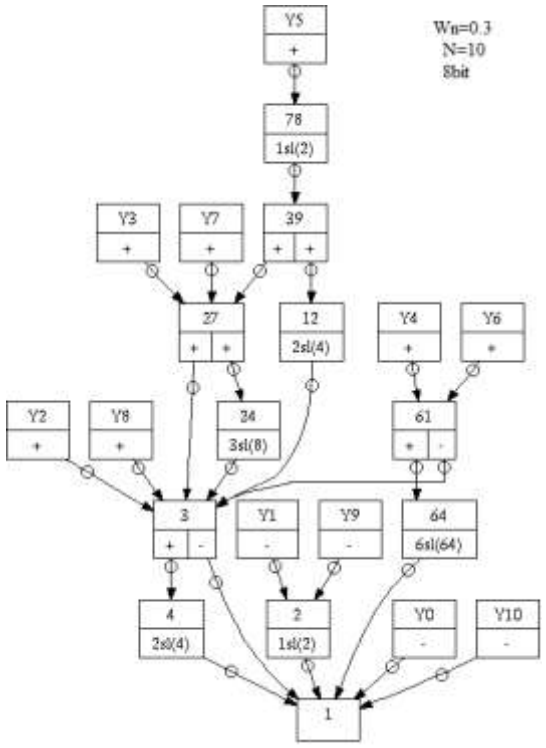


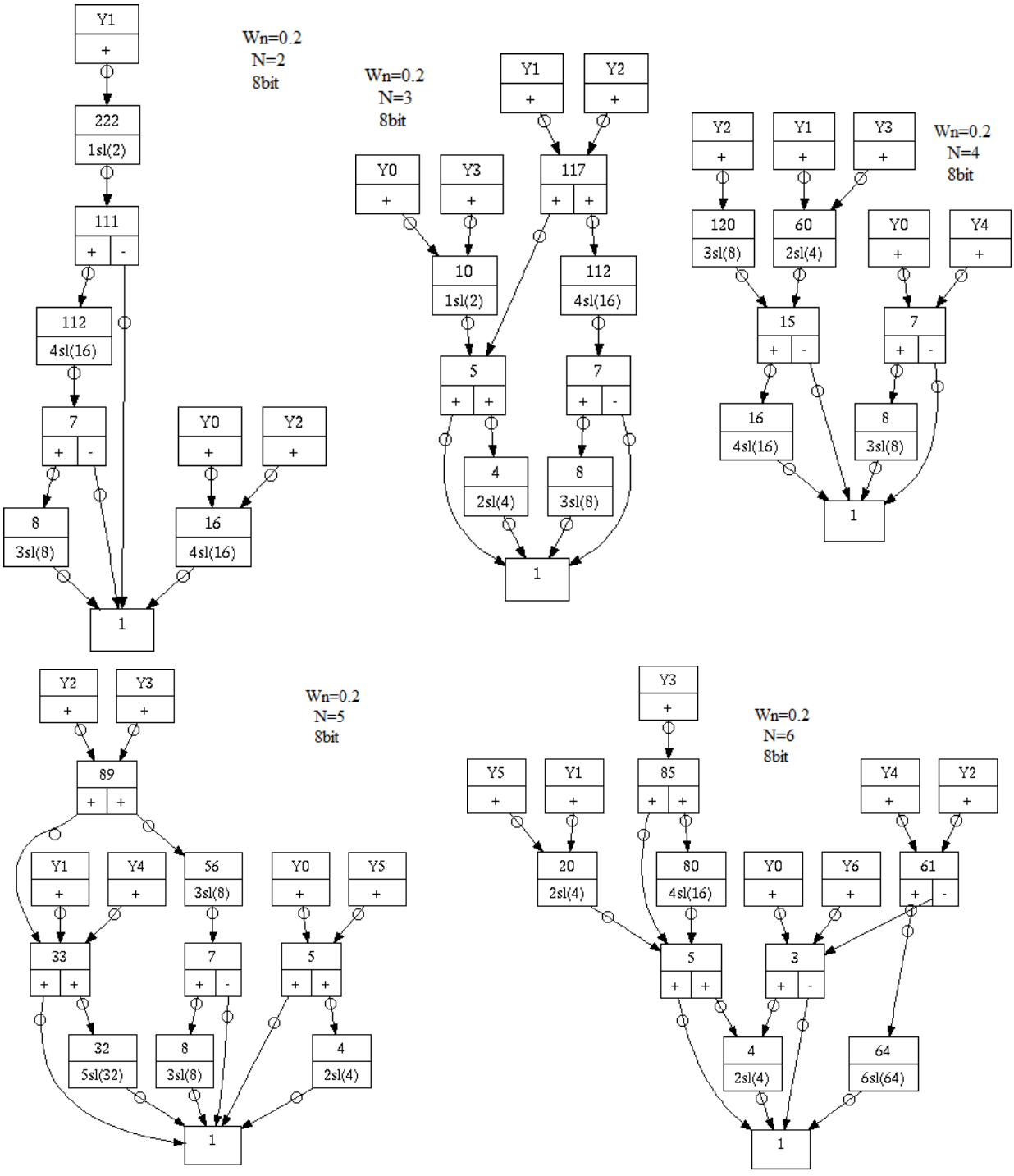
Wn=0.3
N=6
8bit

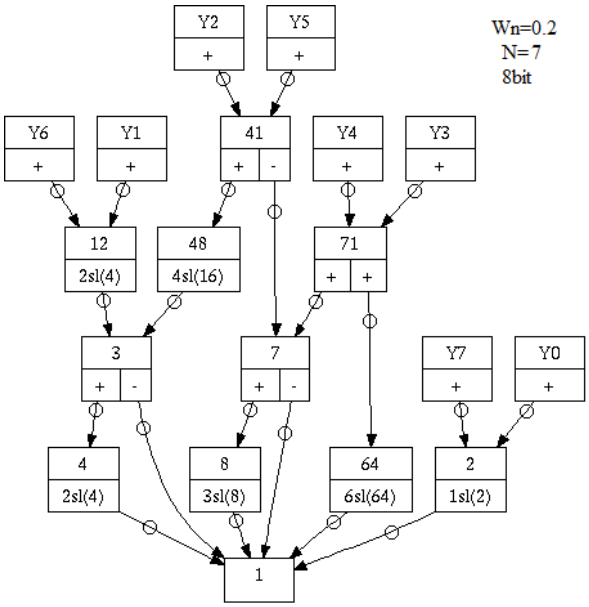

Y1
Y4
Y2
Y3
Wn=0.3
N=7
8bit
Y6
39
81
Y0
Y5
0
40
3sl(8)
80
4sl(16)
6
1sl(2)
5
3
4
2sl(4)
1

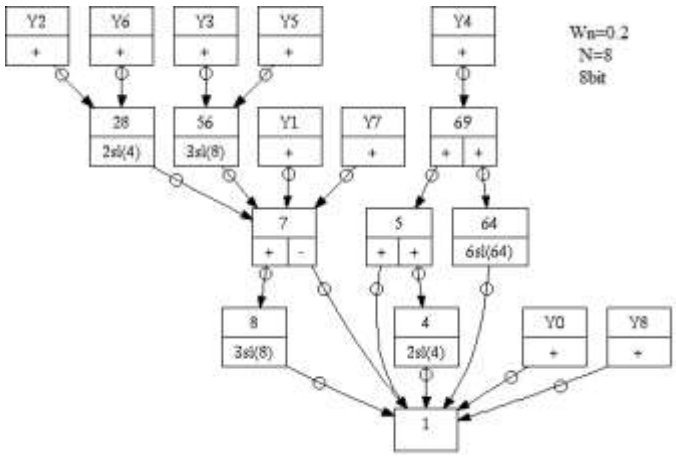

Y6
Y2
Y4
Wn=0.3
N=8
8bit
Y0
22
1sl(2)
83
Y3
Y5
Y1
11
80
4sl(16)
62
1sl(2)
Y7
8
3sl(8)
3
5
31
Y8
4
2sl(4)
32
5sl(32)
1

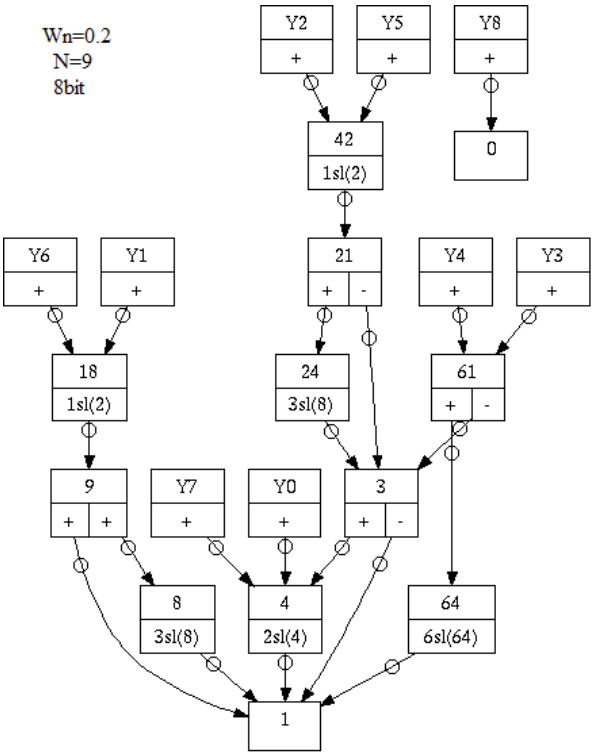

Y3
Y6
Y4
Y5
Y1
Y8
Y2
Y7
43
75
0
Wn=0.3
N=9
8bit
11
80
4sl(16)
16
4sl(16)
32
5sl(32)
5
4
2sl(4)
Y0
Y9
1

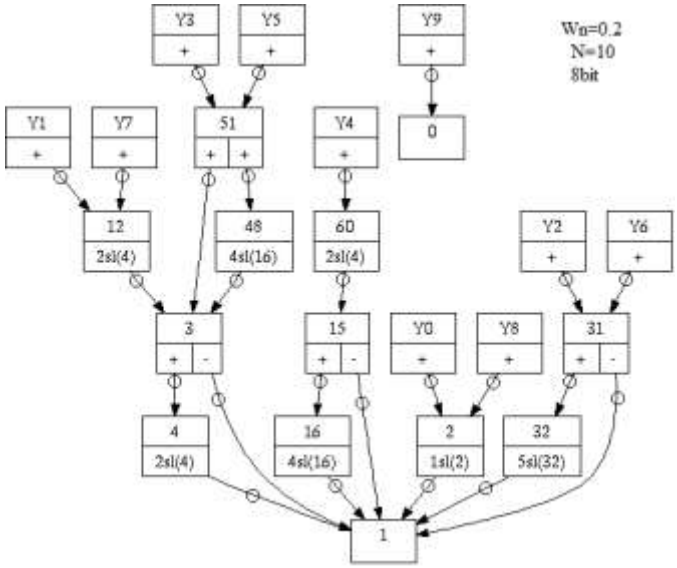

Y5
Wn=0.3
N=10
8bit
78
1sl(2)
Y3
Y7
39
27
12
2sl(4)
Y4
Y6
Y2
Y8
34
3sl(8)
61
3
Y1
Y9
64
6sl(64)
4
2sl(4)
2
1sl(2)
Y0
Y10
1

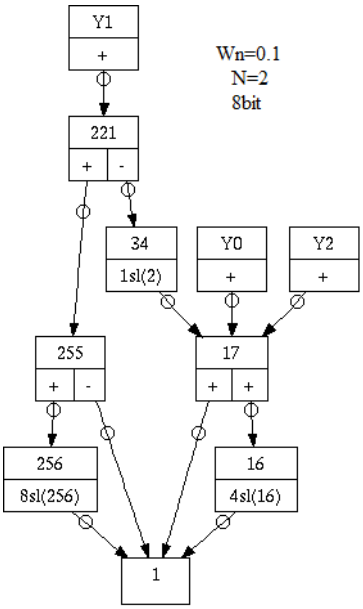

Wn=0.2
N=2
8bit
Y1
+
222
1sl(2)
111
+ -
112
4sl(16)
7
+ -
Y0
+
Y2
+
8
3sl(8)
16
4sl(16)
1
Wn=0.2
N=3
8bit
Y1
+
Y2
+
Y0
+
Y3
+
117
+ +
10
1sl(2)
112
4sl(16)
5
+ +
7
+ -
4
2sl(4)
8
3sl(8)
1
Wn=0.2
N=4
8bit
Y2
+
Y1
+
Y3
+
120
3sl(8)
60
2sl(4)
Y0
+
Y4
+
15
+ -
7
+ -
16
4sl(16)
8
3sl(8)
1
Wn=0.2
N=5
8bit
Y2
+
Y3
+
89
+ +
Y1
+
Y4
+
56
3sl(8)
Y0
+
Y5
+
33
+ +
7
+ -
5
+ +
32
5sl(32)
8
3sl(8)
4
2sl(4)
1
Wn=0.2
N=6
8bit
Y3
+
Y5
+
Y1
+
85
+ +
Y4
+
Y2
+
20
2sl(4)
80
4sl(16)
Y0
+
Y6
+
61
+ -
5
+ +
3
+ -
4
2sl(4)
64
6sl(64)
1

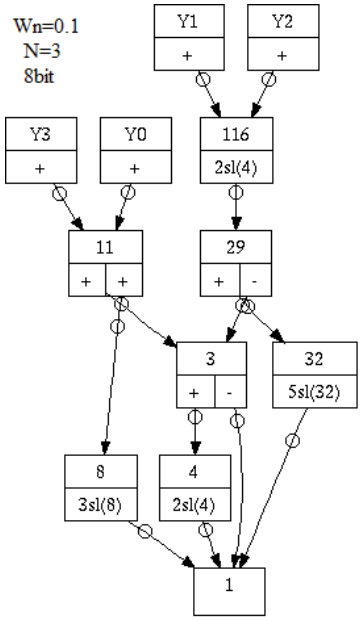

Wn=0.2
N=7
8bit
Y2
Y5
Y6
Y1
41
Y4
Y3
12
2sl(4)
48
4sl(16)
71
3
7
Y7
Y0
4
2sl(4)
8
3sl(8)
64
6sl(64)
2
1sl(2)
1

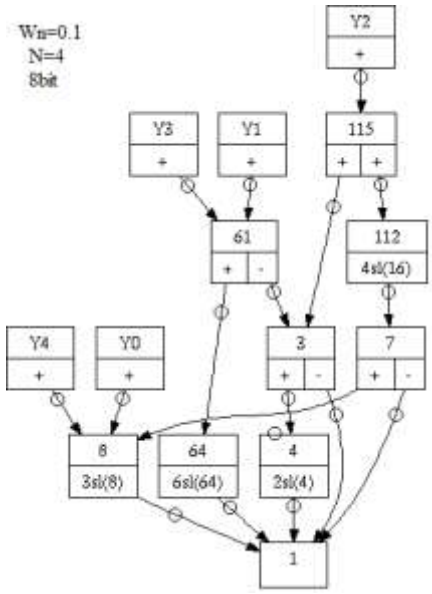

Wn=0.2
N=8
8bit
Y2
Y6
Y3
Y5
Y4
28
2sl(4)
56
3sl(8)
Y1
Y7
69
7
5
64
6sl(64)
8
3sl(8)
4
2sl(4)
Y0
Y8
1

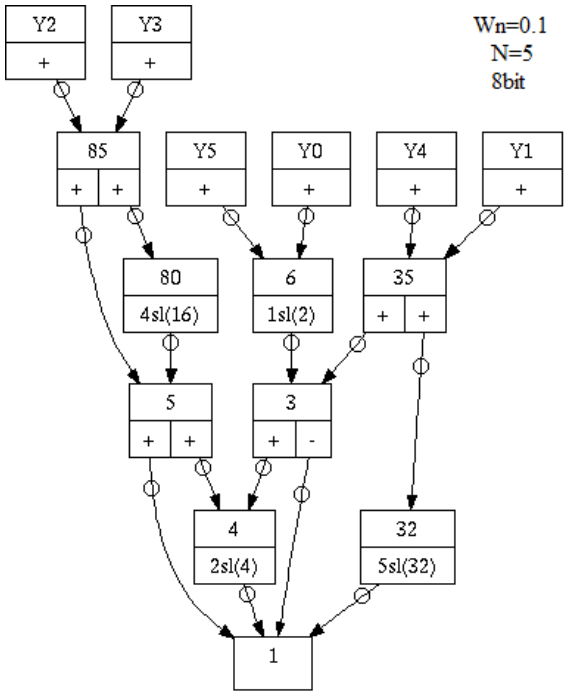

Wn=0.2
N=9
8bit
Y2
Y5
Y8
42
1sl(2)
0
Y6
Y1
21
Y4
Y3
18
1sl(2)
24
3sl(8)
61
9
Y7
Y0
3
8
3sl(8)
4
2sl(4)
64
6sl(64)
1

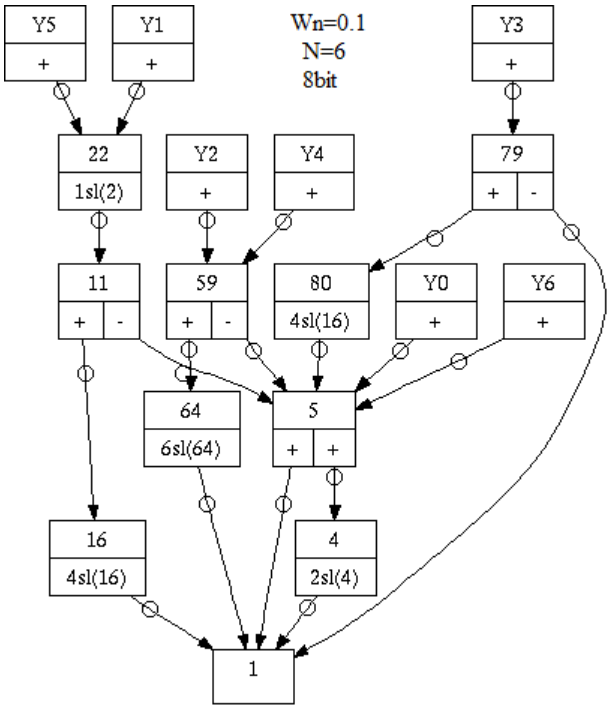

Wn=0.2
N=10
8bit
Y3
Y5
Y9
Y1
Y7
51
Y4
0
12
2sl(4)
48
4sl(16)
60
2sl(4)
Y2
Y6
3
15
Y0
Y8
31
4
2sl(4)
16
4sl(16)
2
1sl(2)
32
5sl(32)
1

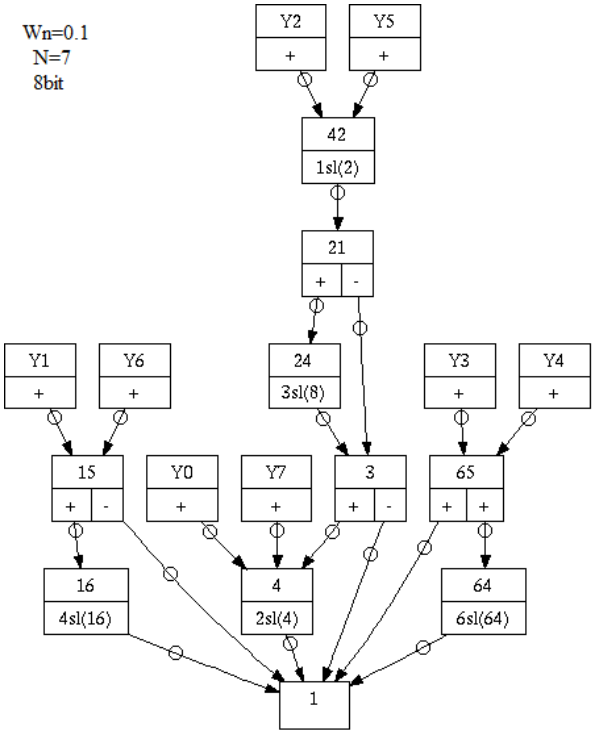

Y1
+
Wn=0.1
N=2
8bit
221
+ -
34
1sl(2)
Y0
+
Y2
+
255
+ -
17
+ +
256
8sl(256)
16
4sl(16)
1

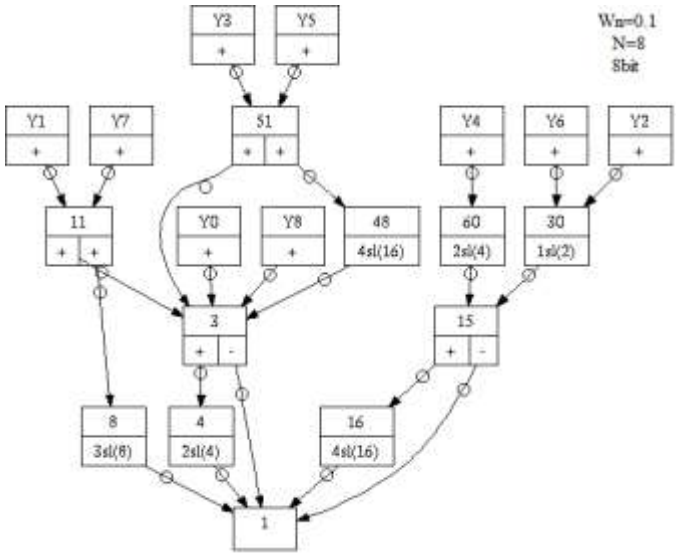

Wn=0.1
N=3
8bit
Y1
+
Y2
+
Y3
+
Y0
+
116
2sl(4)
11
+ +
29
+ -
3
+ -
32
5sl(32)
8
3sl(8)
4
2sl(4)
1

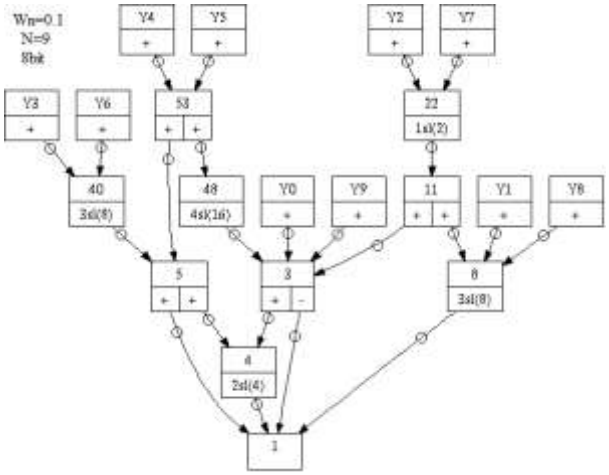

Wn=0.1
N=4
8bit
Y2
+
Y3
+
Y1
+
115
+ +
61
+ -
112
4sl(16)
Y4
+
Y0
+
3
+ -
7
+ -
8
3sl(8)
64
6sl(64)
4
2sl(4)
1

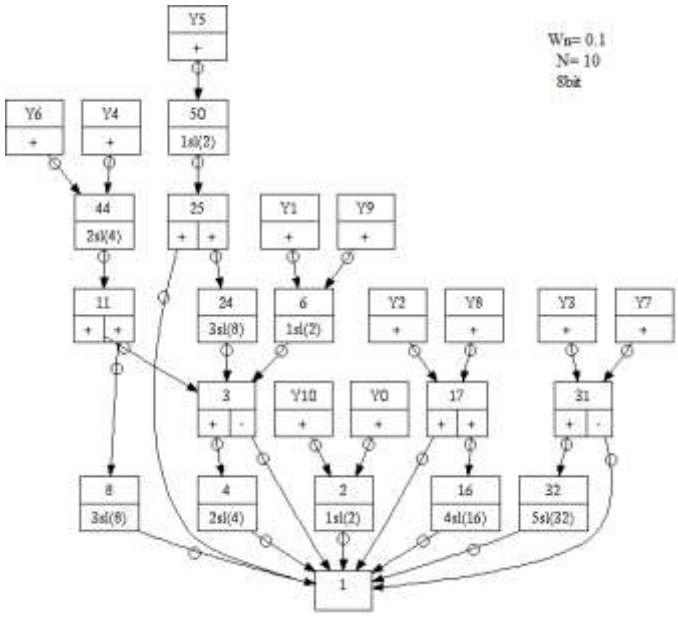

Y2
+
Y3
+
Wn=0.1
N=5
8bit
85
+ +
Y5
+
Y0
+
Y4
+
Y1
+
80
4sl(16)
6
1sl(2)
35
+ +
5
+ +
3
+ -
4
2sl(4)
32
5sl(32)
1


Y5
+
Y1
+
Wn=0.1
N=6
8bit
Y3
+
22
1sl(2)
Y2
+
Y4
+
79
+ -
11
+ -
59
+ -
80
4sl(16)
Y0
+
Y6
+
64
6sl(64)
5
+ +
16
4sl(16)
4
2sl(4)
1

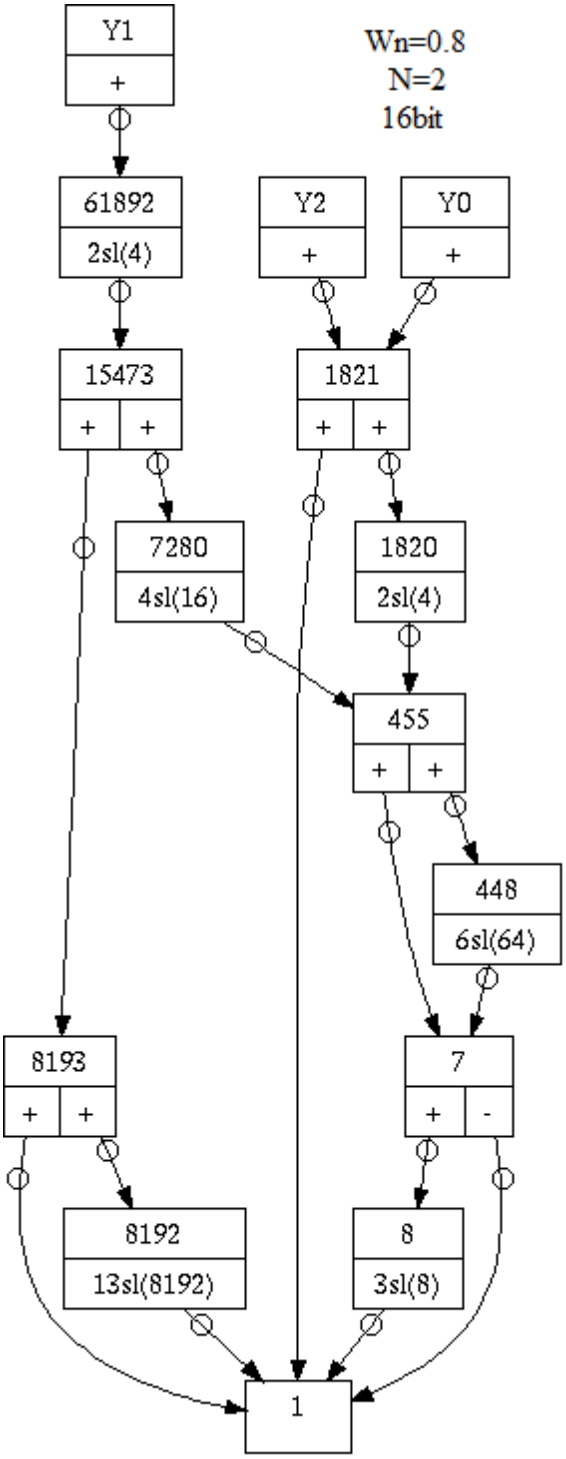

Wn=0.1
N=7
8bit
Y2
Y5
42
1sl(2)
21
Y1
Y6
24
3sl(8)
Y3
Y4
15
Y0
Y7
3
65
16
4sl(16)
4
2sl(4)
64
6sl(64)
1

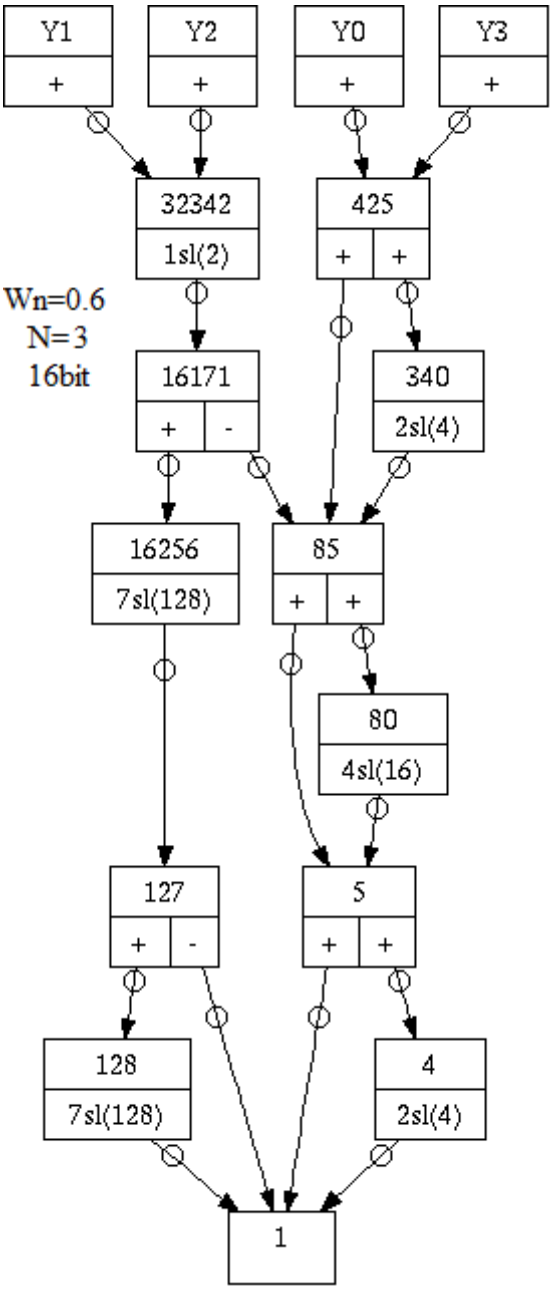

Wn=0.1
N=8
8bit
Y3
Y5
Y1
Y7
51
Y4
Y6
Y2
11
Y0
Y8
48
4sl(16)
60
2sl(4)
30
1sl(2)
3
15
8
3sl(8)
4
2sl(4)
16
4sl(16)
1

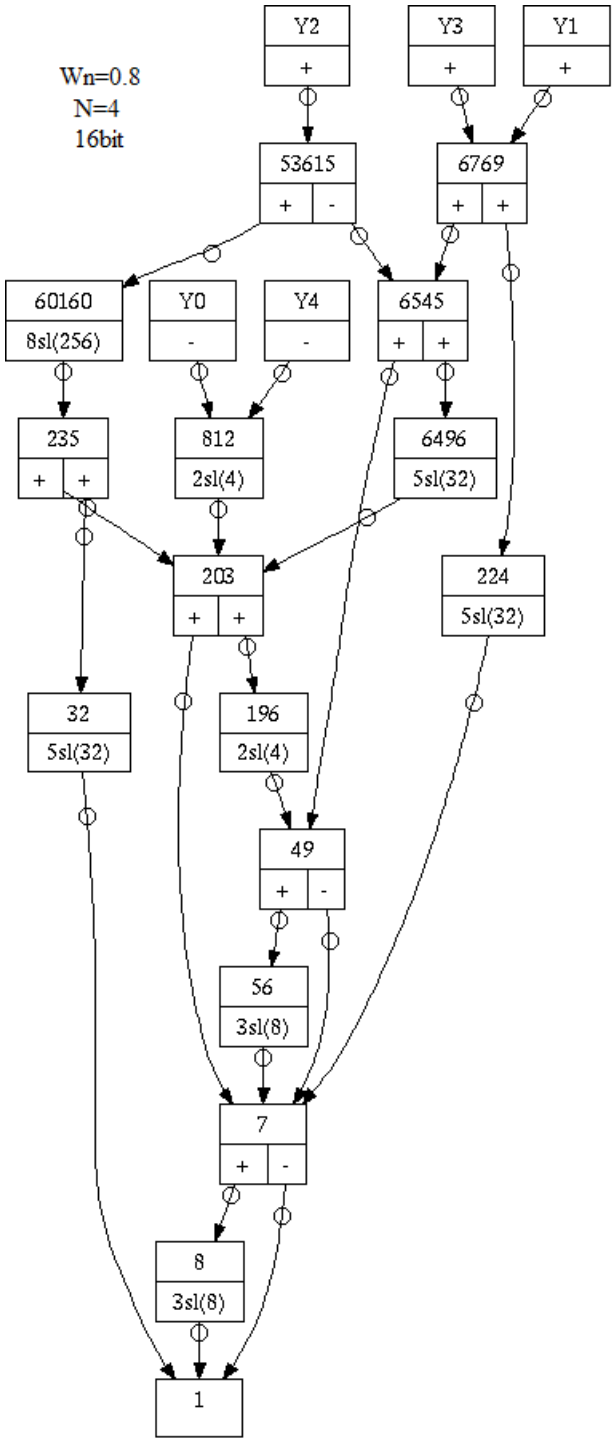

Wn=0.1
N=9
8bit
Y4
Y5
Y2
Y7
Y3
Y6
53
22
1sl(2)
40
3sl(8)
48
4sl(16)
Y0
Y9
11
Y1
Y8
5
3
8
3sl(8)
4
2sl(4)
1

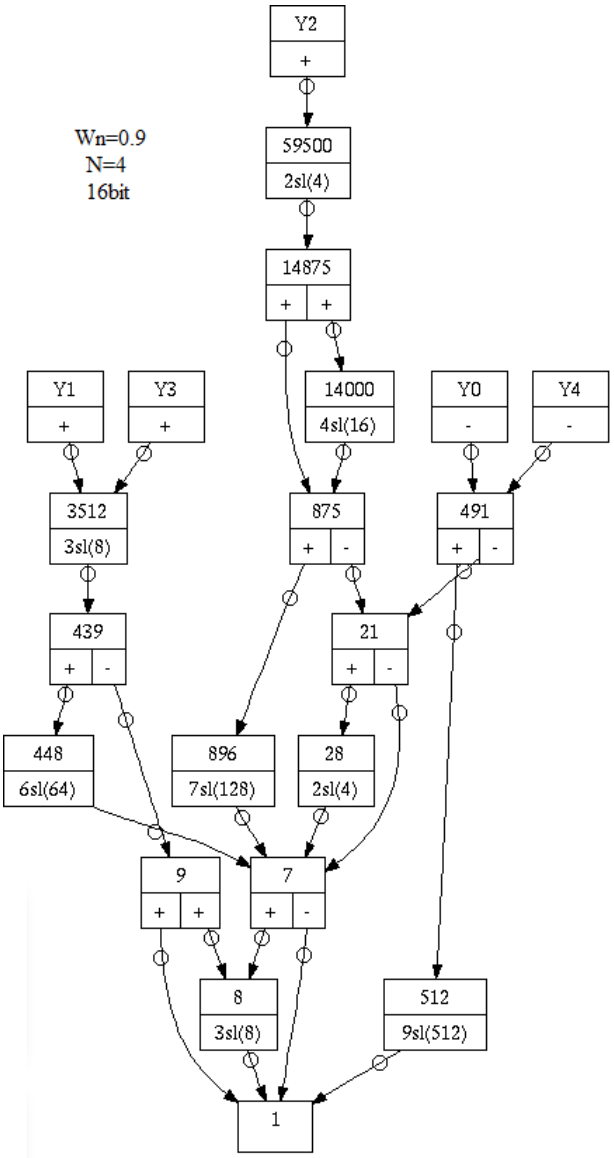

Wn= 0.1
N= 10
8bit
Y5
Y6
Y4
50
1sl(2)
44
2sl(4)
25
Y1
Y9
11
24
3sl(8)
6
1sl(2)
Y2
Y8
Y3
Y7
3
Y10
Y0
17
31
8
3sl(8)
4
2sl(4)
2
1sl(2)
16
4sl(16)
32
5sl(32)
1

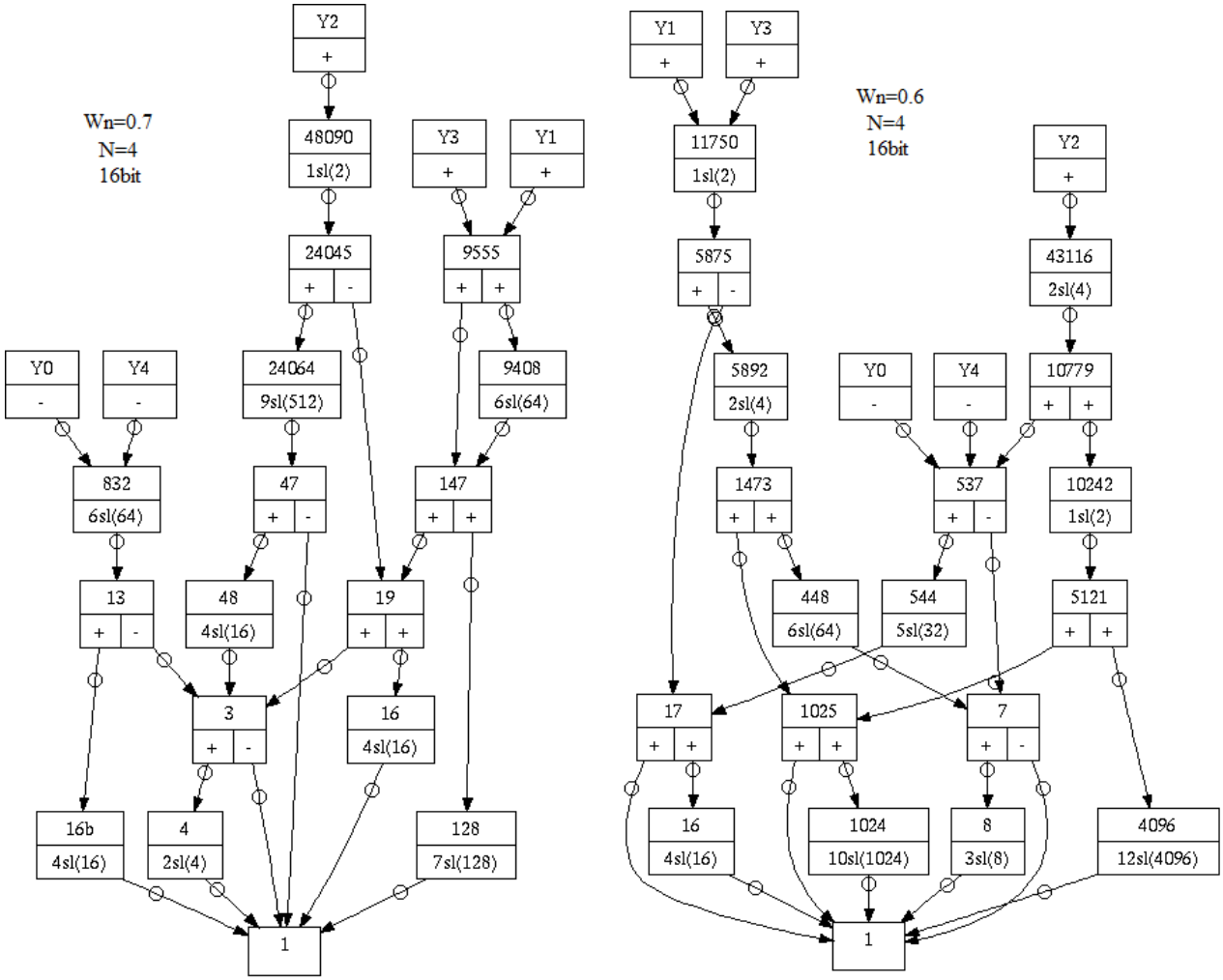
A-6.2 Grafos de Filtros FIR Spiral Paso bajo 16 bit de precisión.



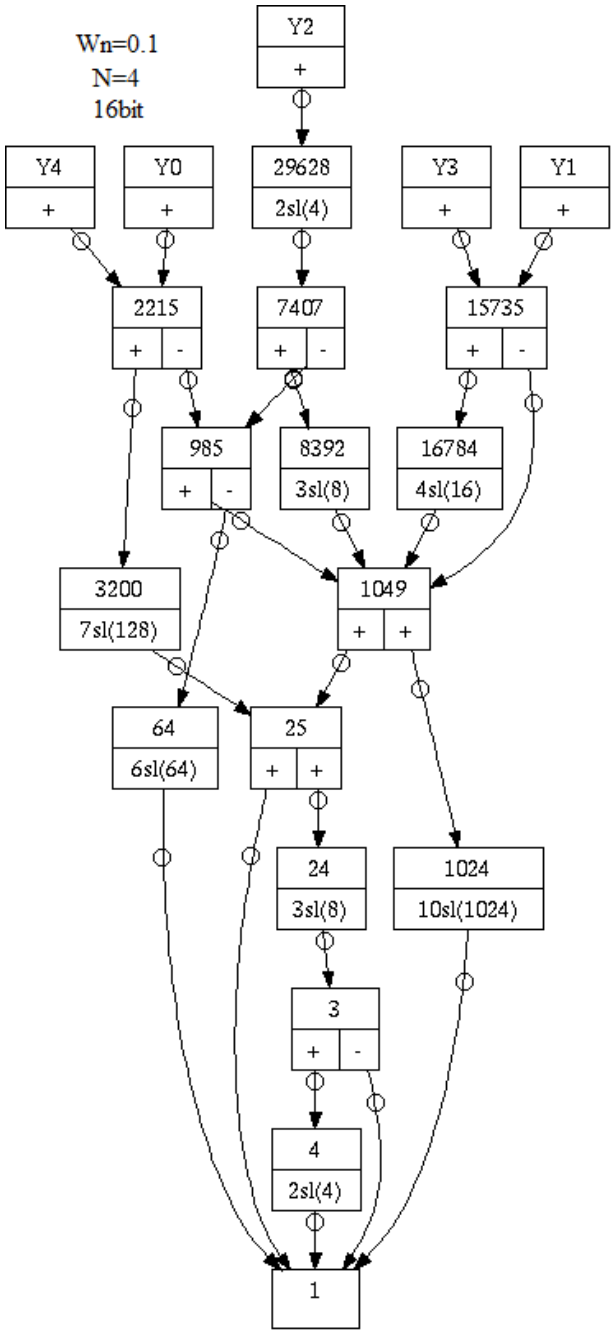


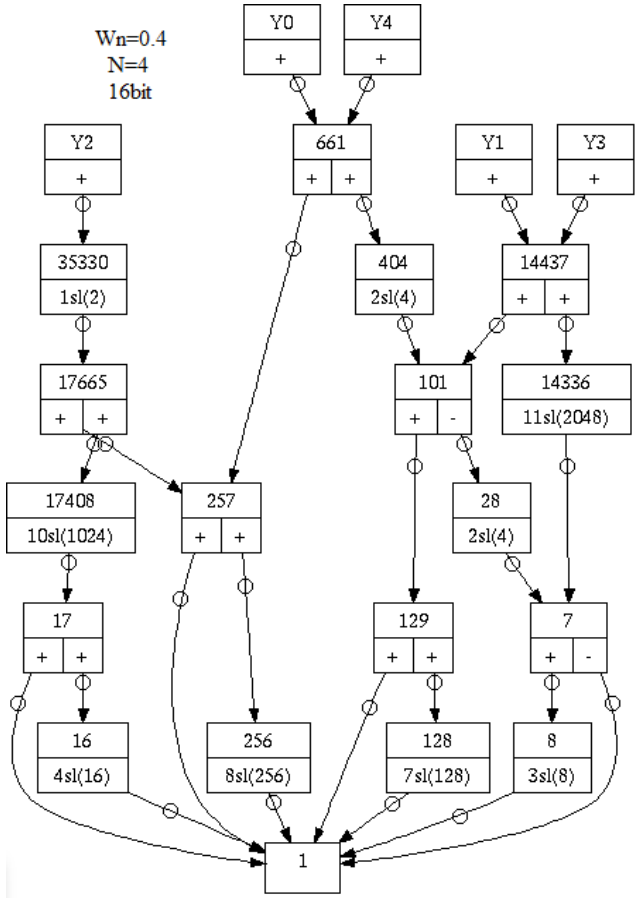

Wn=0.8
N=4
16bit
Y2
+
Y3
+
Y1
+
53615
+ -
6769
+ +
60160
8sl(256)
Y0
-
Y4
-
6545
+ +
235
+ +
812
2sl(4)
6496
5sl(32)
203
+ +
224
5sl(32)
32
5sl(32)
196
2sl(4)
49
+ -
56
3sl(8)
7
+ -
8
3sl(8)
1

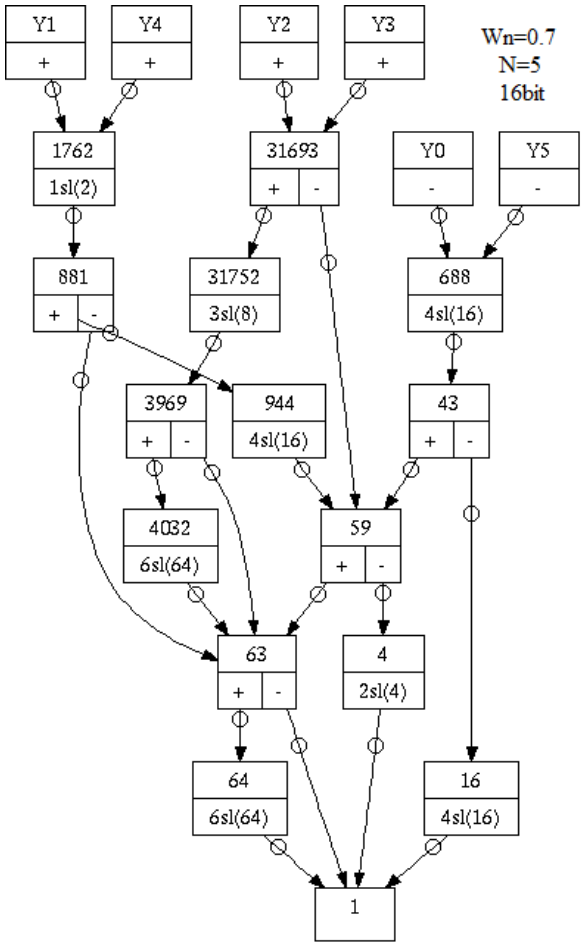

Wn=0.9
N=4
16bit
Y2
+
59500
2sl(4)
14875
+ +
Y1
+
Y3
+
14000
4sl(16)
Y0
-
Y4
-
3512
3sl(8)
875
+ -
491
+ -
439
+ -
21
+ -
448
6sl(64)
896
7sl(128)
28
2sl(4)
9
+ +
7
+ -
8
3sl(8)
512
9sl(512)
1

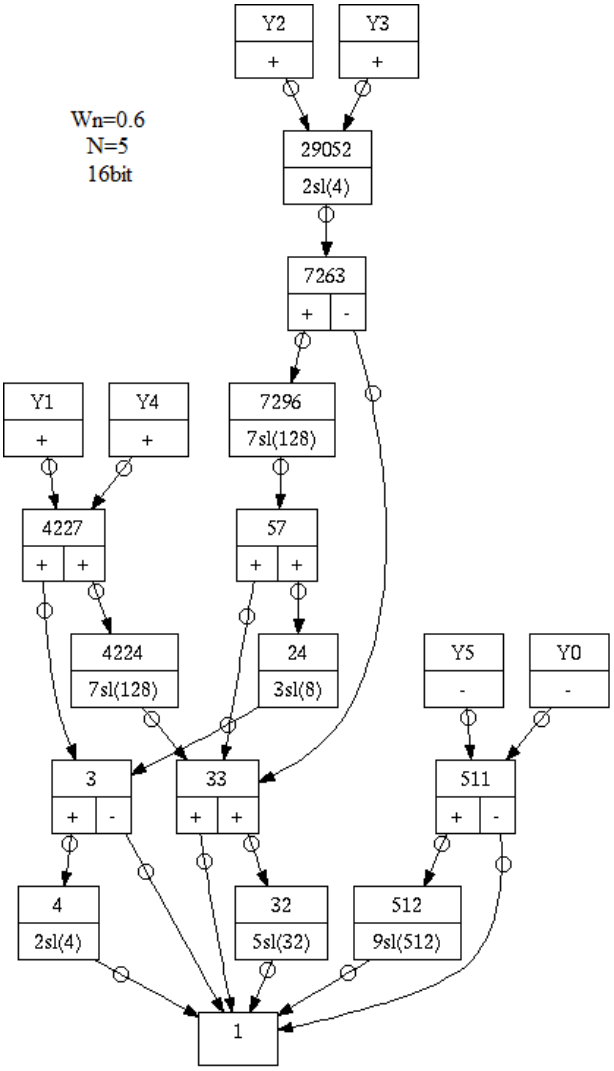

Wn=0.7
N=4
16bit
Y2
+
48090
1sl(2)
Y3
+
Y1
+
24045
+ -
9555
+ +
Y0
-
Y4
-
24064
9sl(512)
9408
6sl(64)
832
6sl(64)
47
+ -
147
+ +
13
+ -
48
4sl(16)
19
+ +
3
+ -
16
4sl(16)
16b
4sl(16)
4
2sl(4)
128
7sl(128)
1
Wn=0.6
N=4
16bit
Y1
+
Y3
+
11750
1sl(2)
5875
+ -
Y2
+
43116
2sl(4)
5892
2sl(4)
Y0
-
Y4
-
10779
+ +
1473
+ +
537
+ -
10242
1sl(2)
448
6sl(64)
544
5sl(32)
5121
+ +
17
+ +
1025
+ +
7
+ -
16
4sl(16)
1024
10sl(1024)
8
3sl(8)
4096
12sl(4096)
1

f

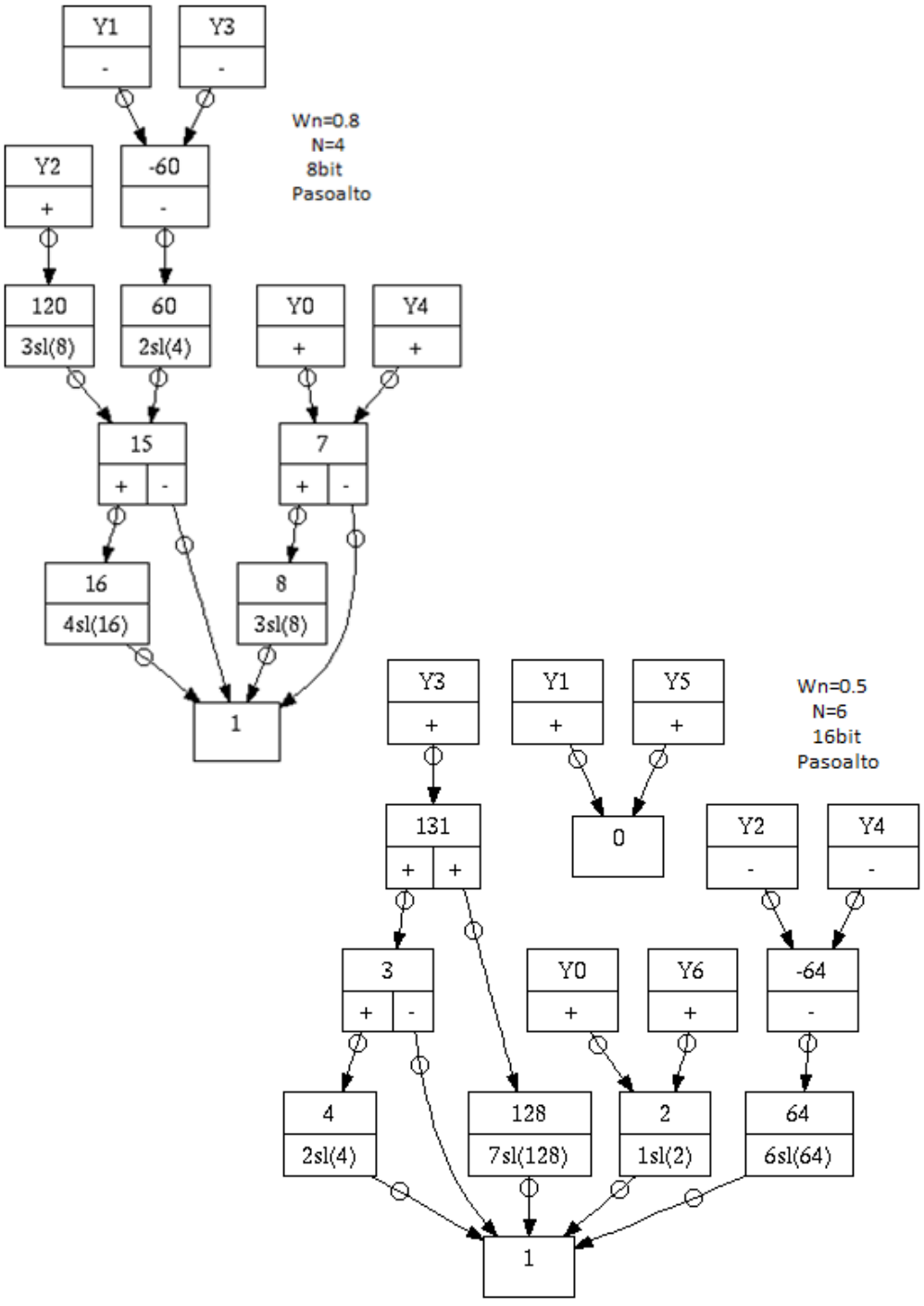

Wn=0.1
N=4
16bit
Y2
+
Y4
+
Y0
+
29628
2sl(4)
Y3
+
Y1
+
2215
+ -
7407
+ -
15735
+ -
985
+ -
8392
3sl(8)
16784
4sl(16)
3200
7sl(128)
1049
+ +
64
6sl(64)
25
+ +
24
3sl(8)
1024
10sl(1024)
3
+ -
4
2sl(4)
1


Wn=0.4
N=4
16bit
Y0
+
Y4
+
Y2
+
661
+ +
Y1
+
Y3
+
35330
1sl(2)
404
2sl(4)
14437
+ +
17665
+ +
101
+ -
14336
11sl(2048)
17408
10sl(1024)
257
+ +
28
2sl(4)
17
+ +
129
+ +
7
+ -
16
4sl(16)
256
8sl(256)
128
7sl(128)
8
3sl(8)
1


Y1
+
Y4
+
Y2
+
Y3
+
Wn=0.7
N=5
16bit
1762
1sl(2)
31693
+ -
Y0
-
Y5
-
881
+ -
31752
3sl(8)
688
4sl(16)
3969
+ -
944
4sl(16)
43
+ -
4032
6sl(64)
59
+ -
63
+ -
4
2sl(4)
64
6sl(64)
16
4sl(16)
1

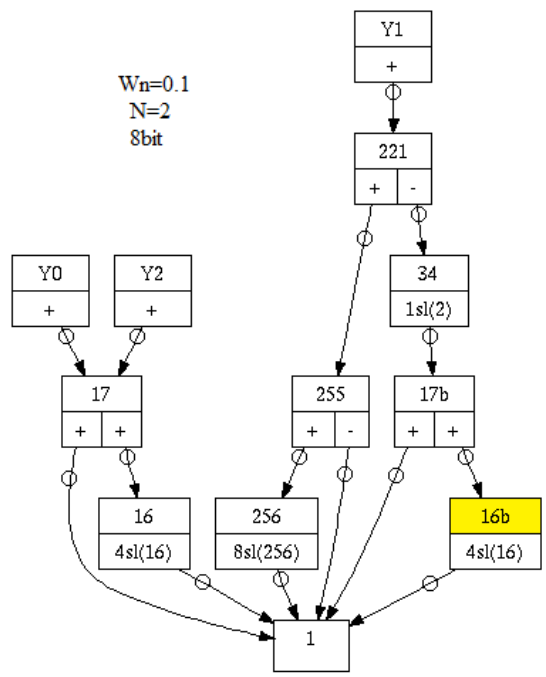

Y2
+
Y3
+
Wn=0.6
N=5
16bit
29052
2sl(4)
7263
+ -
Y1
+
Y4
+
7296
7sl(128)
4227
+ +
57
+ +
4224
7sl(128)
24
3sl(8)
Y5
-
Y0
-
3
+ -
33
+ +
511
+ -
4
2sl(4)
32
5sl(32)
512
9sl(512)
1

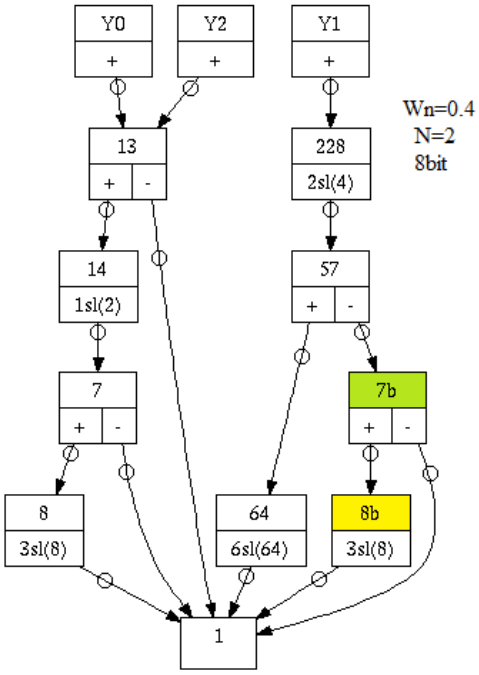
A-6.3 Grafos de Filtros FIR Spiral Paso alto.

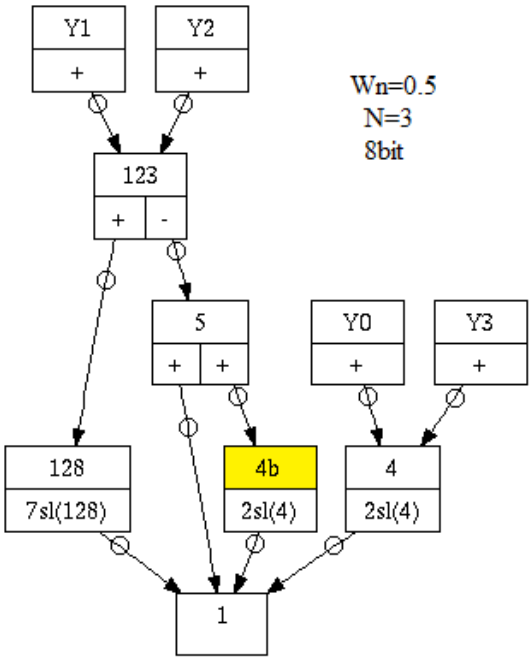


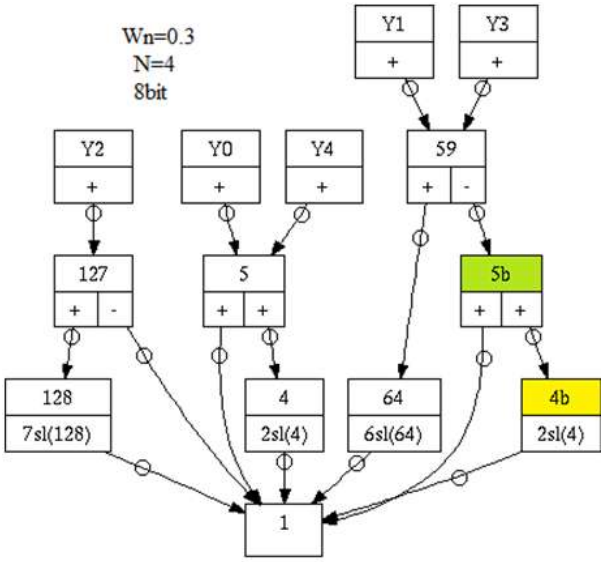
Anexo A-7 Grafos de Filtros FIR Paso bajo con nodos replicados

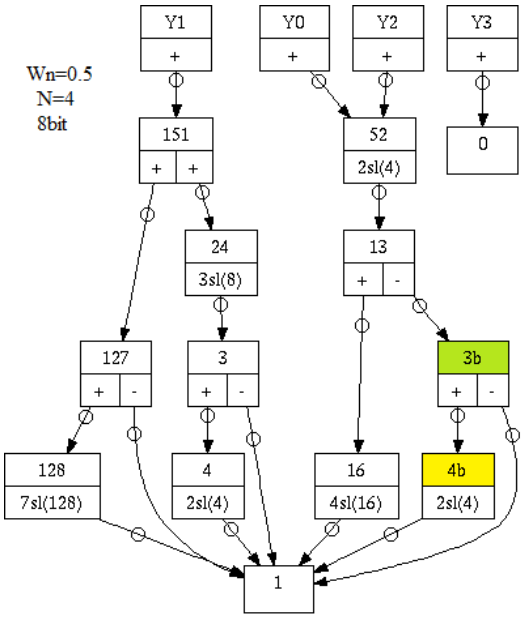
A-7.1 Filtros con nodos replicados de 8 bit de precisión.

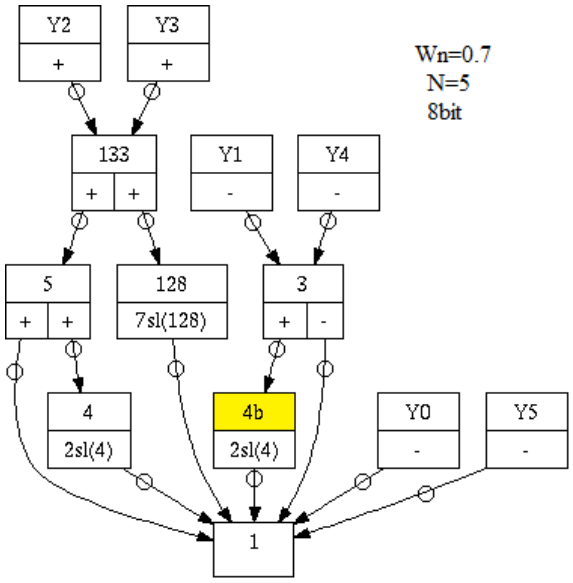


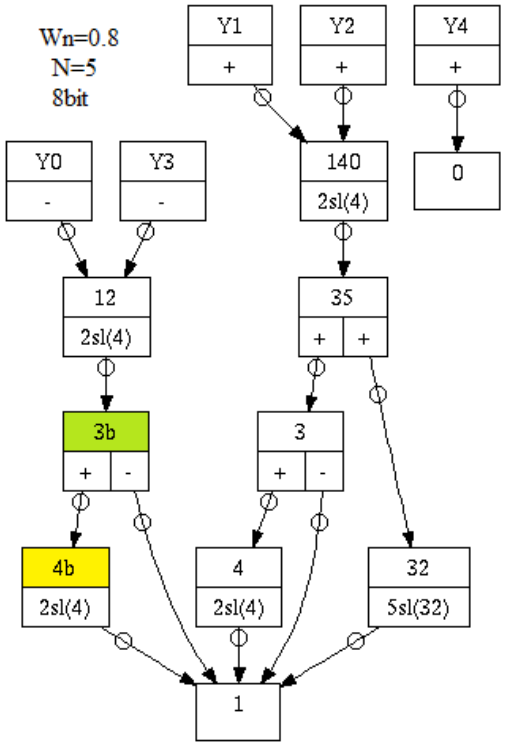


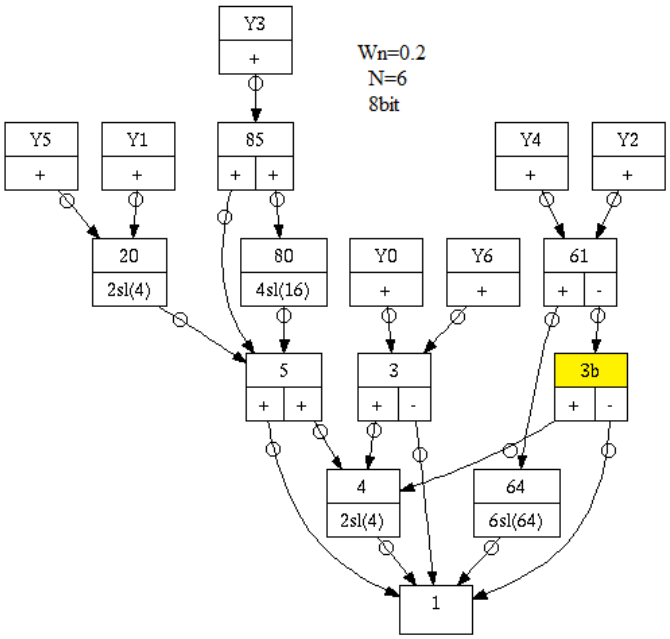


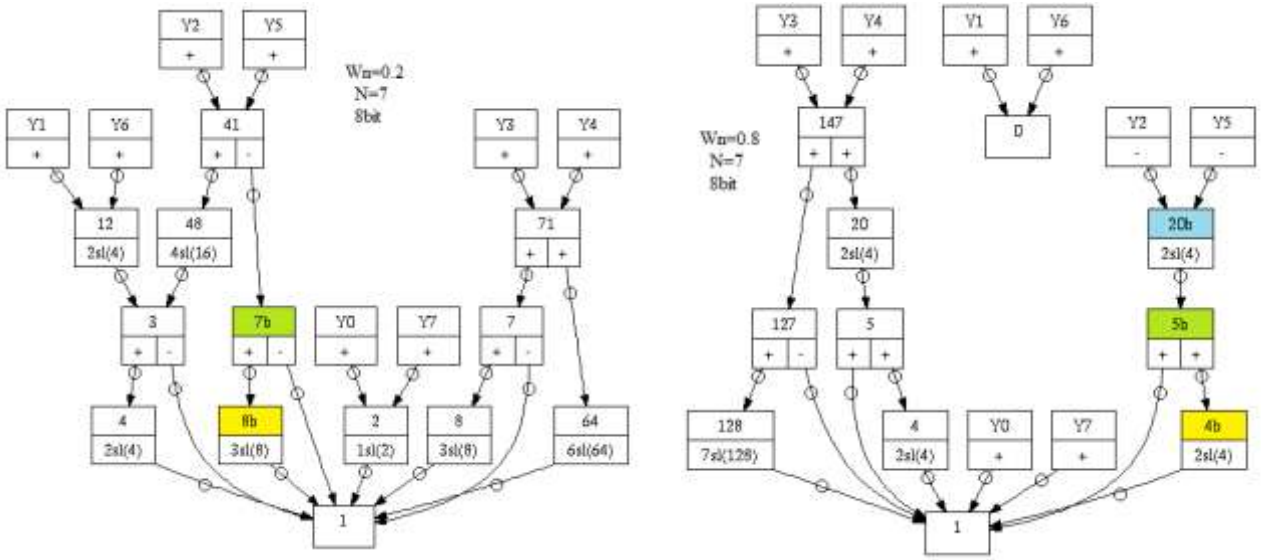


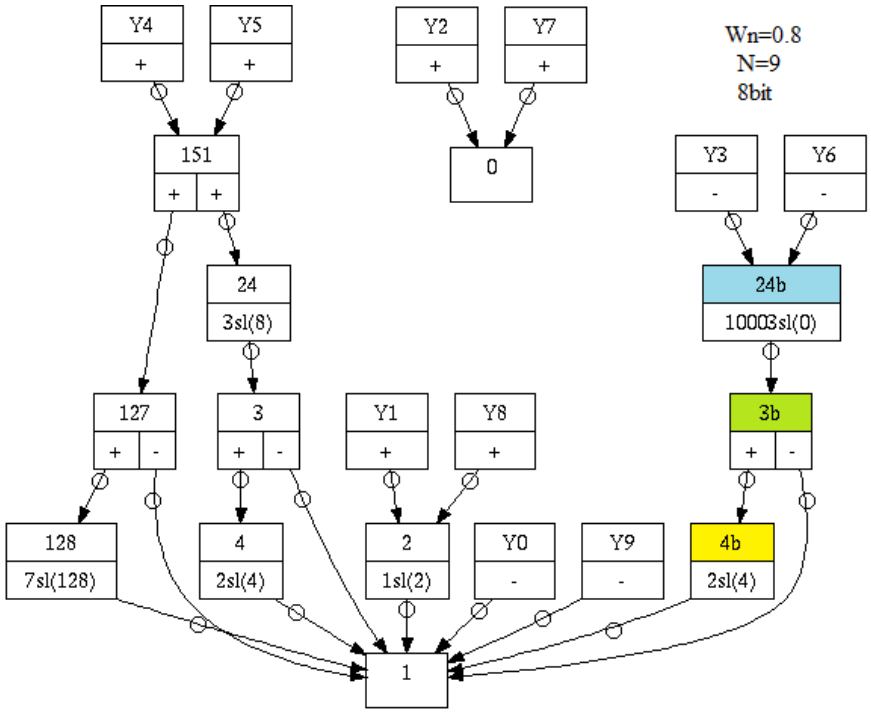

Wn=0.5
N=4
8bit
Y1
Y0
Y2
Y3
151
52
2sl(4)
0
24
3sl(8)
13
127
3
3b
128
7sl(128)
4
2sl(4)
16
4sl(16)
4b
2sl(4)
1

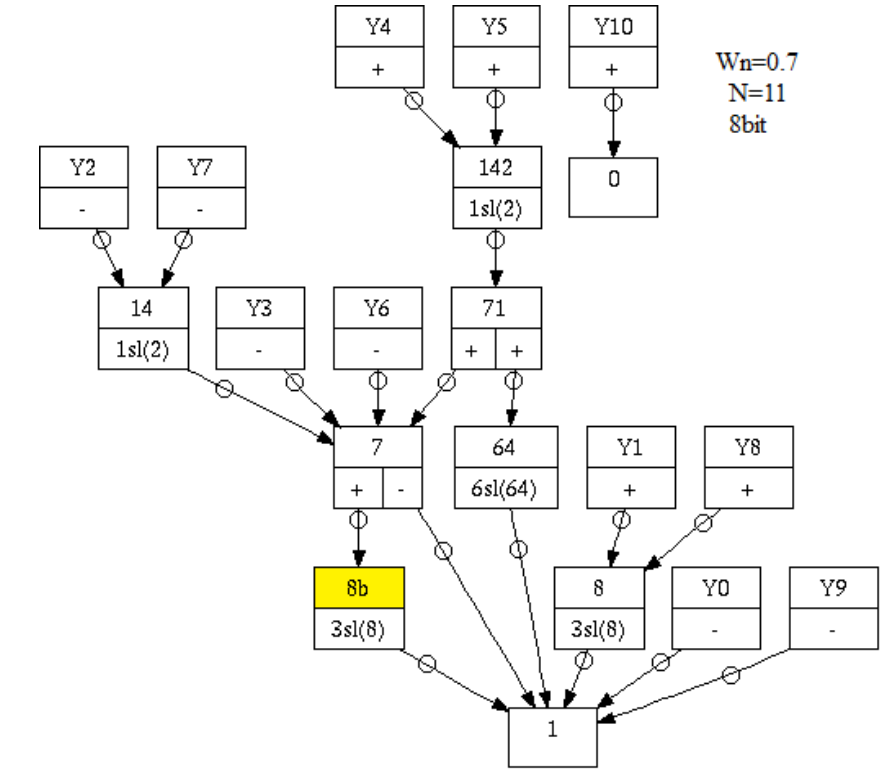

Wn=0.7
N=5
8bit
Y2
Y3
133
Y1
Y4
5
128
7sl(128)
3
4
2sl(4)
4b
2sl(4)
Y0
Y5
1

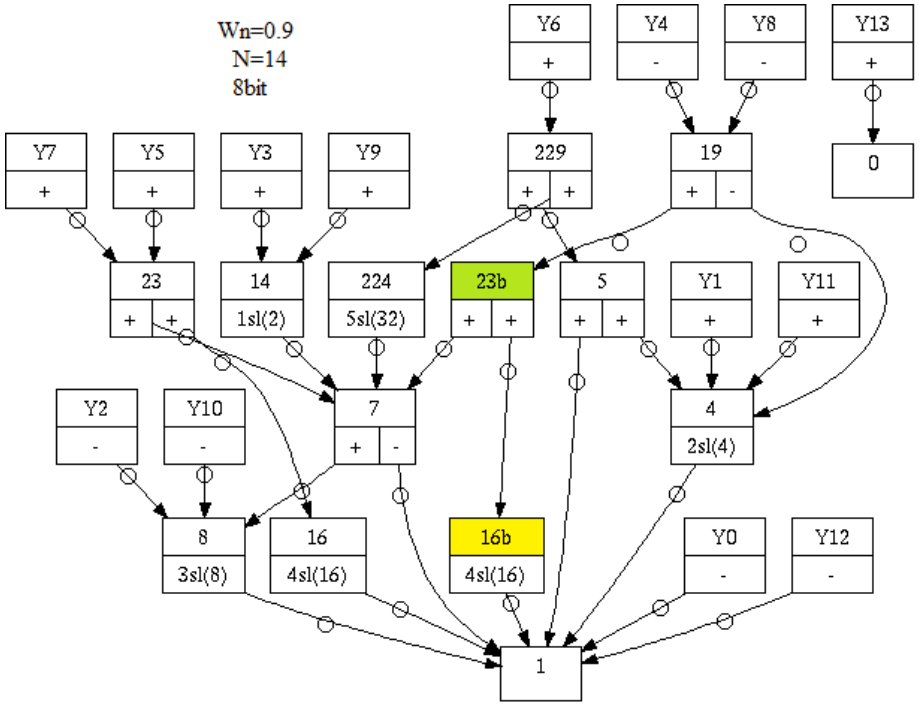

Wn=0.8
N=5
8bit
Y1
Y2
Y4
Y0
Y3
140
2sl(4)
0
12
2sl(4)
35
3b
3
4b
2sl(4)
4
2sl(4)
32
5sl(32)
1

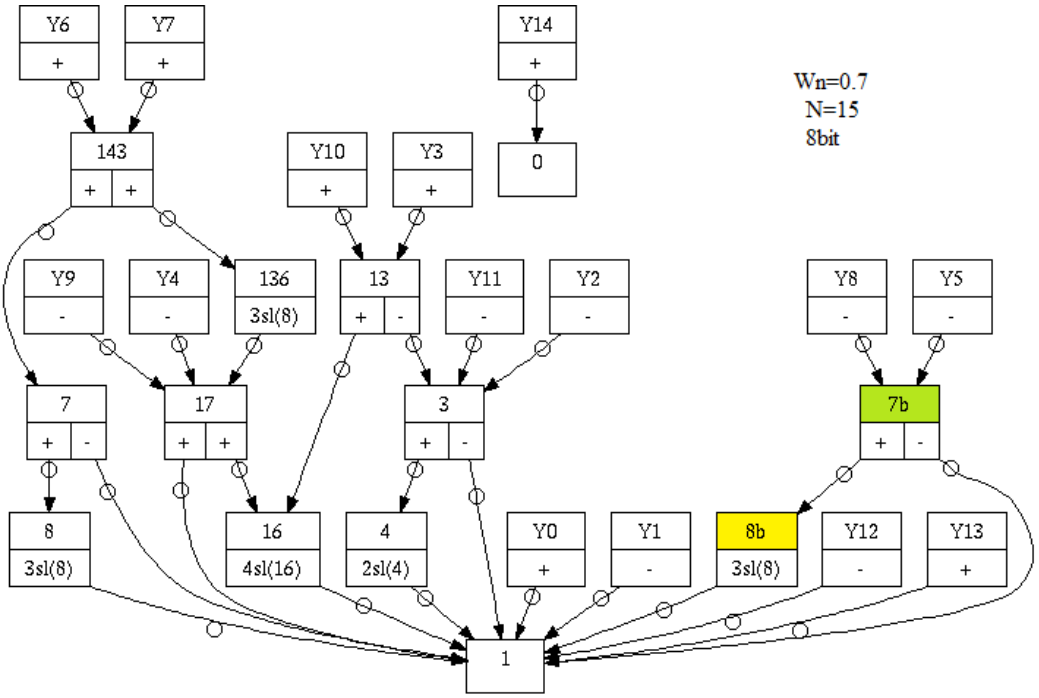

Wn=0.2
N=6
8bit
Y3
Y5
Y1
85
Y4
Y2
20
2sl(4)
80
4sl(16)
Y0
Y6
61
5
3
3b
4
2sl(4)
64
6sl(64)
1

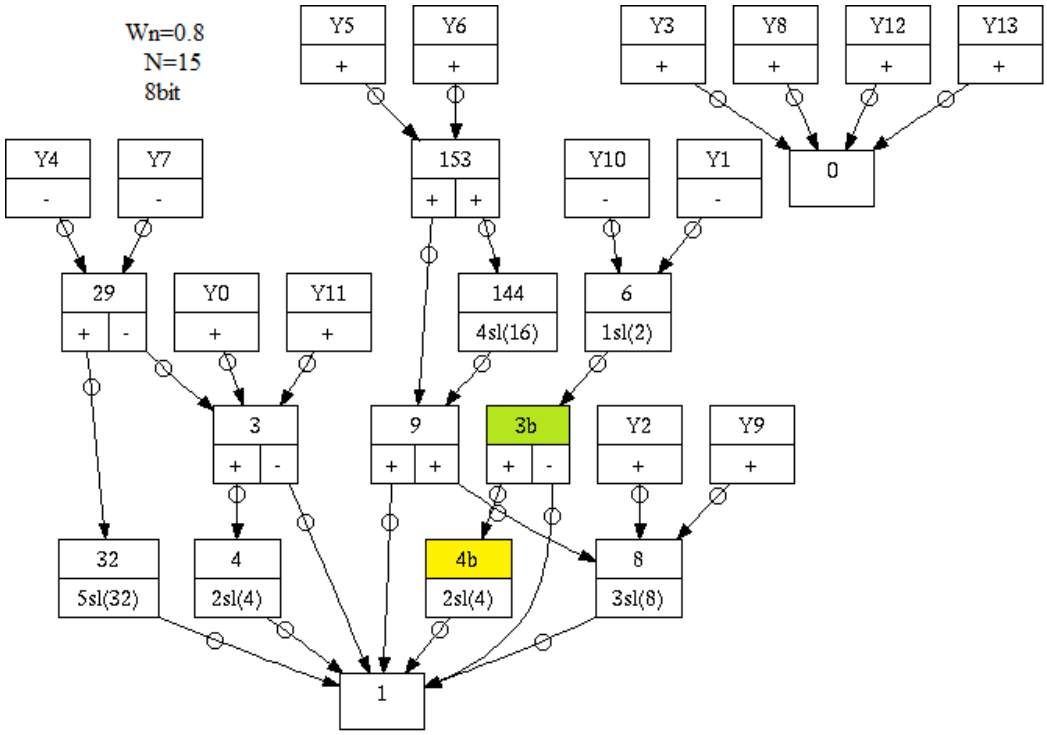

Y2
Y5
Wn=0.2
N=7
8bit
Y1
Y6
41
Y3
Y4
12
2sl(4)
48
4sl(16)
71
3
7b
Y0
Y7
7
4
2sl(4)
8b
3sl(8)
2
1sl(2)
8
3sl(8)
64
6sl(64)
1
Y3
Y4
Y1
Y6
Wn=0.8
N=7
8bit
147
0
Y2
Y5
20
2sl(4)
20b
2sl(4)
127
5
5b
128
7sl(128)
4
2sl(4)
Y0
Y7
4b
2sl(4)
1

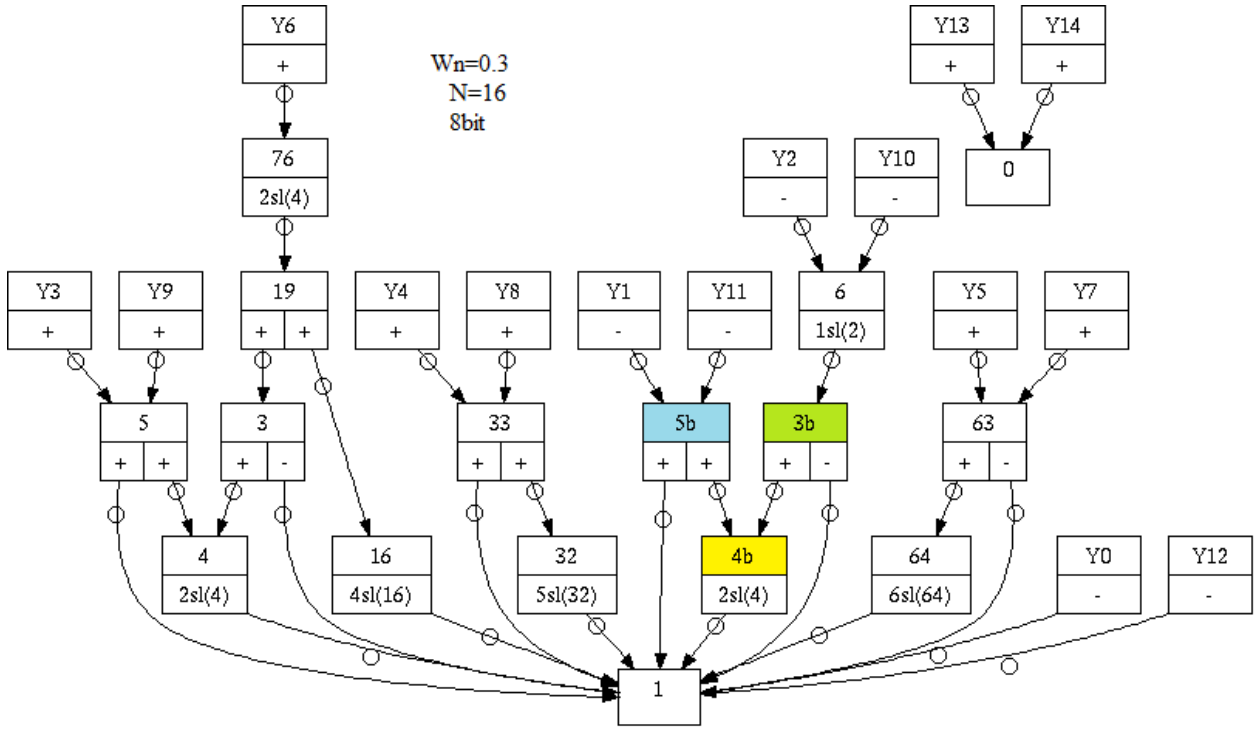

Y4
Y5
Y2
Y7
Wn=0.8
N=9
8bit
151
0
Y3
Y6
24
3sl(8)
24b
10003sl(0)
127
3
Y1
Y8
3b
128
7sl(128)
4
2sl(4)
2
1sl(2)
Y0
Y9
4b
2sl(4)
1

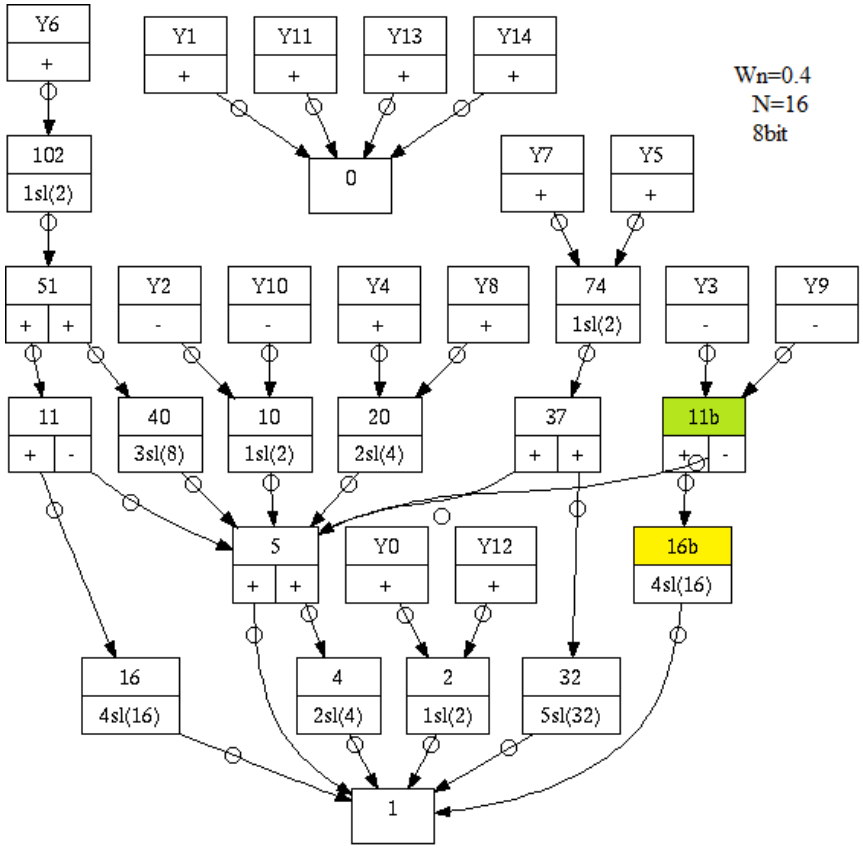

Y4
+
Y5
+
Y10
+
Wn=0.7
N=11
8bit
Y2
-
Y7
-
142
1sl(2)
0
14
1sl(2)
Y3
-
Y6
-
71
+
+
7
+
-
64
6sl(64)
Y1
+
Y8
+
8b
3sl(8)
8
3sl(8)
Y0
-
Y9
-
1


Wn=0.9
N=14
8bit
Y6
+
Y4
-
Y8
-
Y13
+
Y7
+
Y5
+
Y3
+
Y9
+
229
+
+
19
+
-
0
23
+
+
14
1sl(2)
224
5sl(32)
23b
+
+
5
+
+
Y1
+
Y11
+
Y2
-
Y10
-
7
+
-
4
2sl(4)
8
3sl(8)
16
4sl(16)
16b
4sl(16)
Y0
-
Y12
-
1

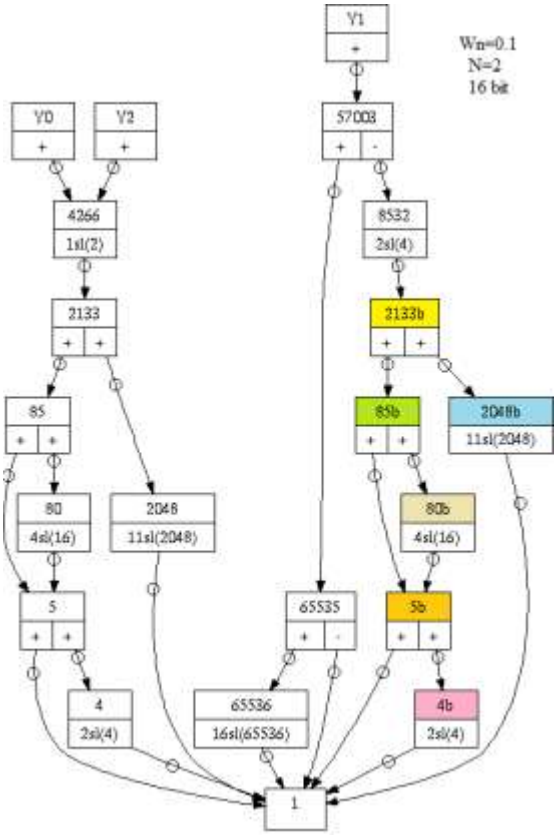

Y6
Y7
Y14
Wn=0.7
N=15
8bit
143
Y10
Y3
0
Y9
Y4
136
3sl(8)
13
Y11
Y2
Y8
Y5
7
17
3
7b
8
3sl(8)
16
4sl(16)
4
2sl(4)
Y0
Y1
8b
3sl(8)
Y12
Y13
1

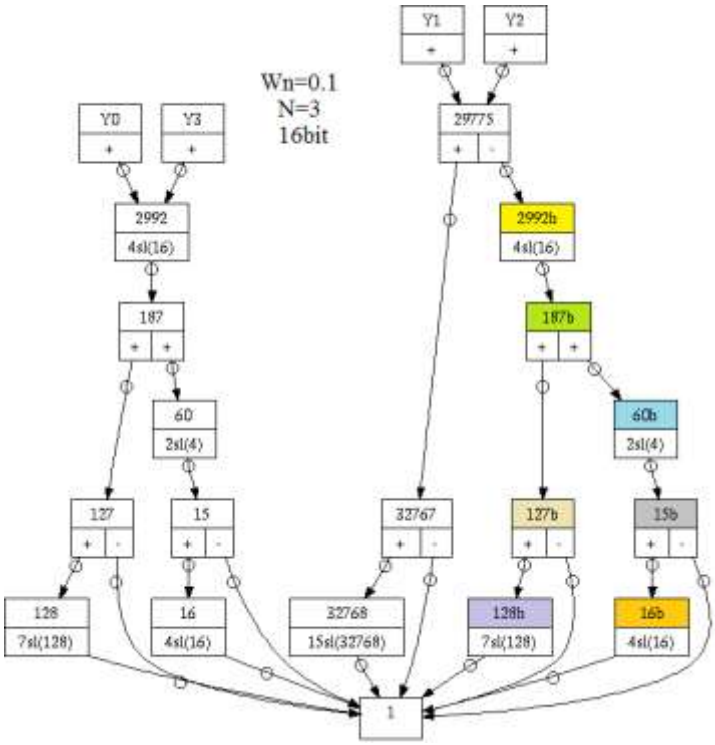

Wn=0.8
N=15
8bit
Y5
Y6
Y3
Y8
Y12
Y13
Y4
Y7
153
Y10
Y1
0
29
Y0
Y11
144
4sl(16)
6
1sl(2)
3
9
3b
Y2
Y9
32
5sl(32)
4
2sl(4)
4b
2sl(4)
8
3sl(8)
1

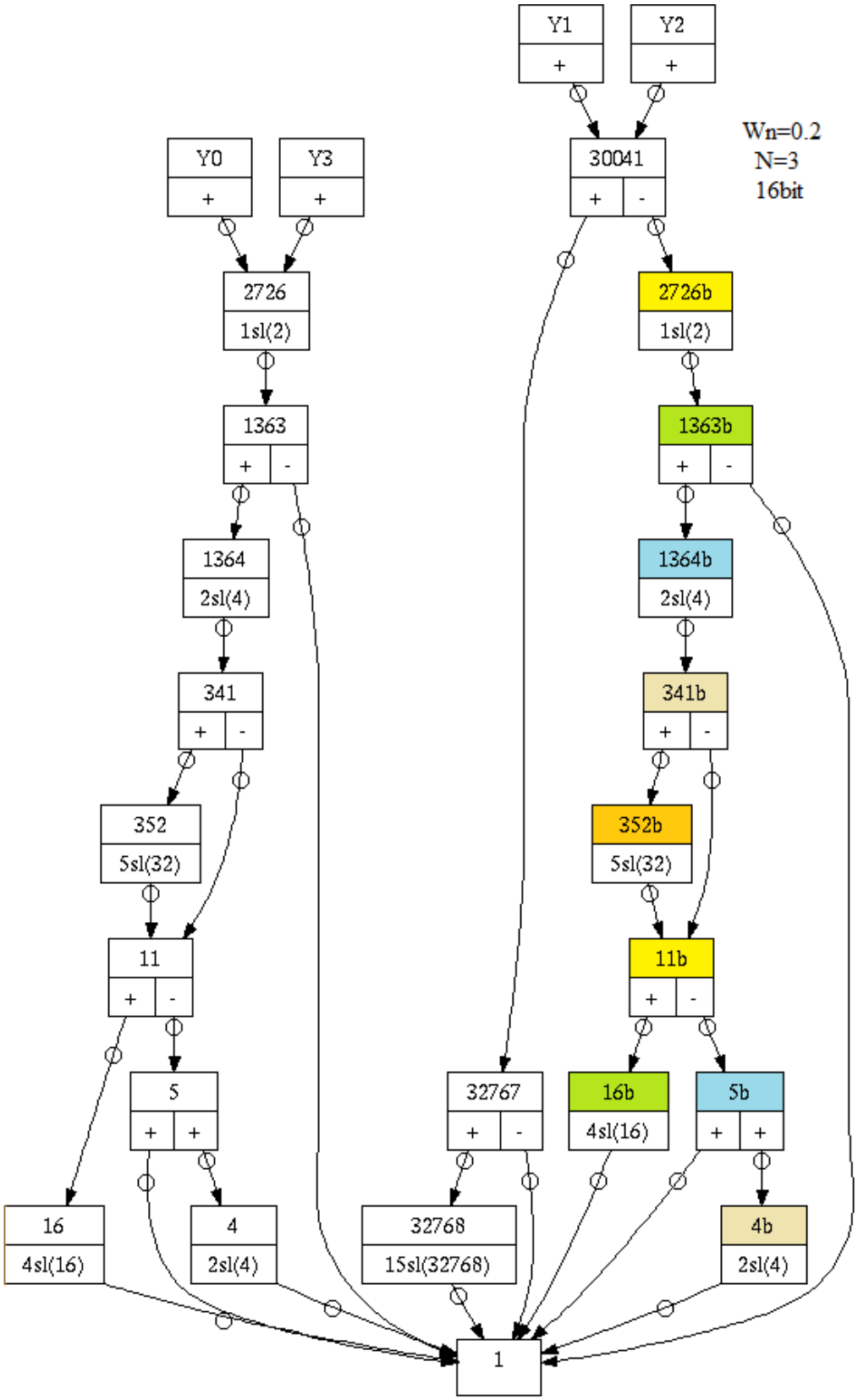

Wn=0.3
N=16
8bit
Y6
+
76
2sl(4)
Y13
+
Y14
+
0
Y2
-
Y10
-
Y3
+
Y9
+
19
+
+
Y4
+
Y8
+
Y1
-
Y11
-
6
1sl(2)
Y5
+
Y7
+
5
+
+
3
+
-
33
+
+
5b
+
+
3b
+
-
63
+
-
4
2sl(4)
16
4sl(16)
32
5sl(32)
4b
2sl(4)
64
6sl(64)
Y0
-
Y12
-
1

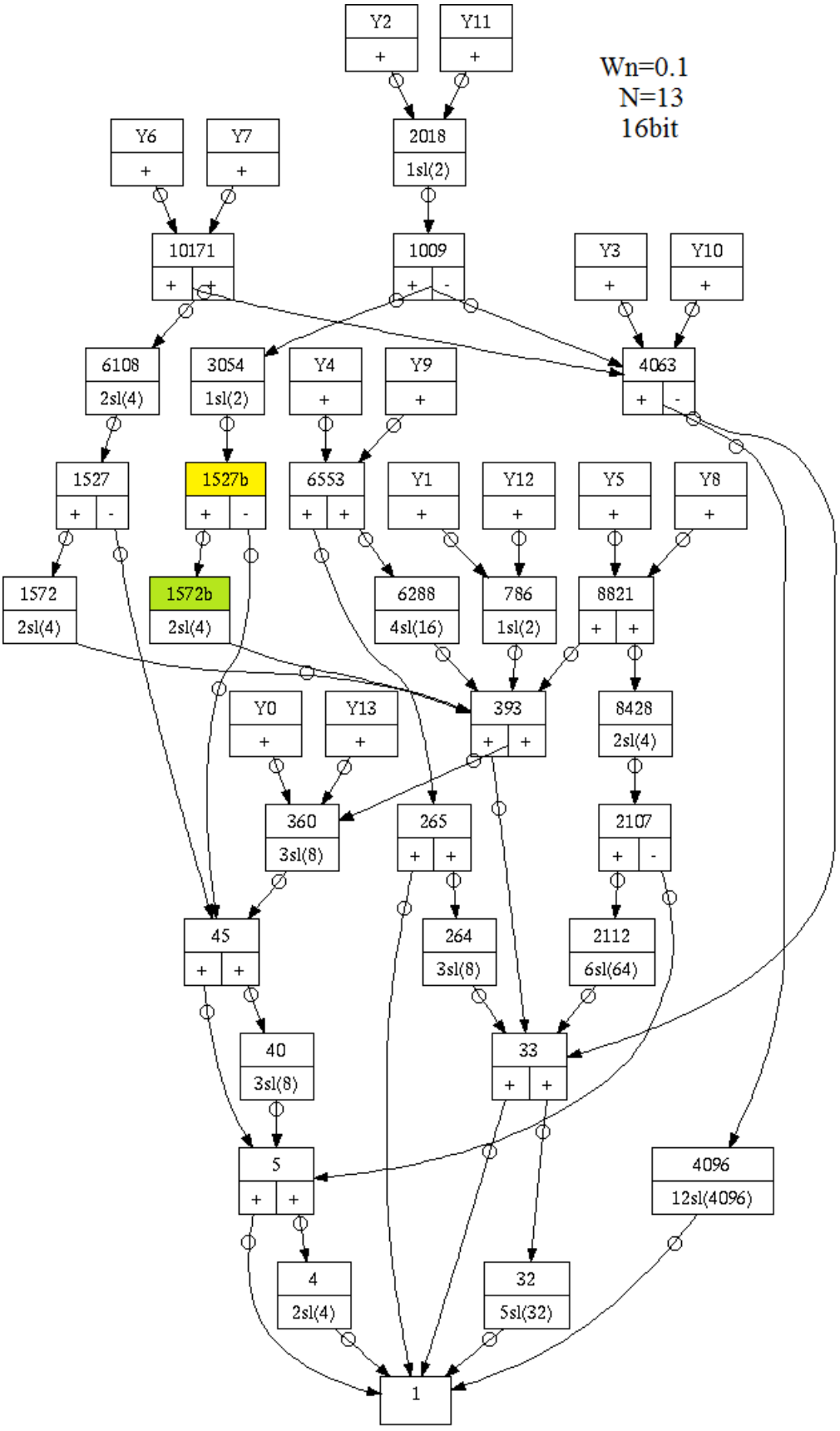

Wn=0.4
N=16
8bit
Y6
+
Y1
+
Y11
+
Y13
+
Y14
+
0
102
1sl(2)
Y7
+
Y5
+
51
+
+
Y2
-
Y10
-
Y4
+
Y8
+
74
1sl(2)
Y3
-
Y9
-
11
+
-
40
3sl(8)
10
1sl(2)
20
2sl(4)
37
+
+
11b
+
-
5
+
+
Y0
+
Y12
+
16b
4sl(16)
16
4sl(16)
4
2sl(4)
2
1sl(2)
32
5sl(32)
1

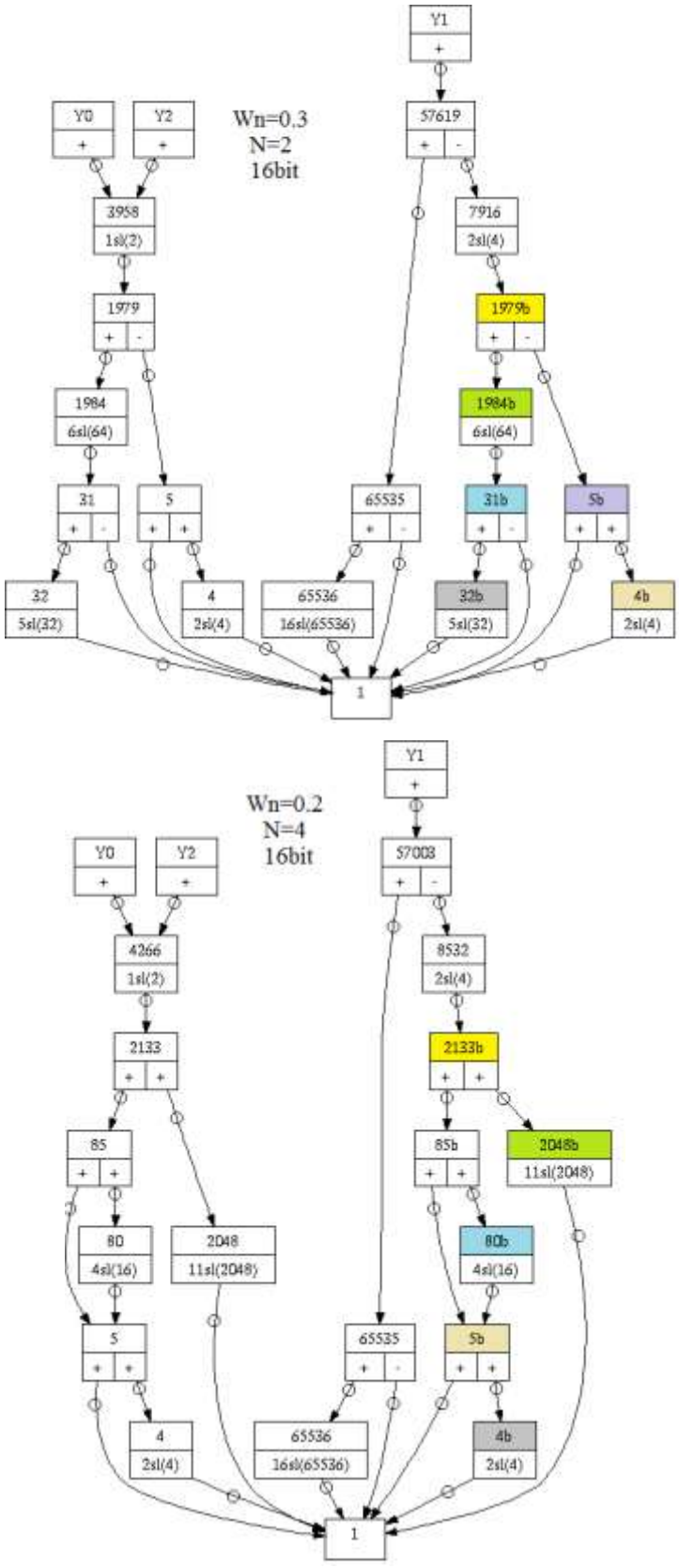
A-7.2 Filtros con nodos replicados de 16 bit de precisión.

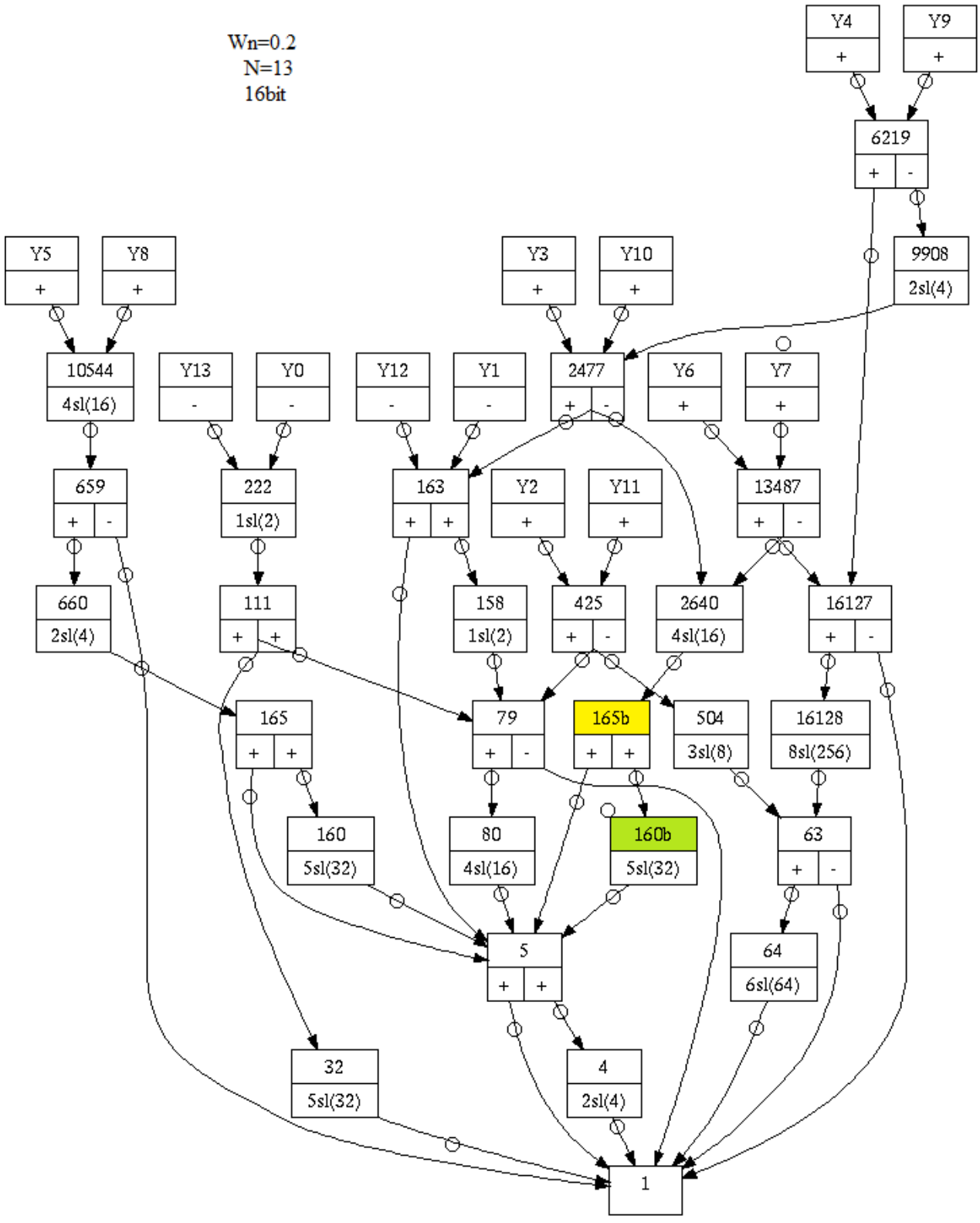


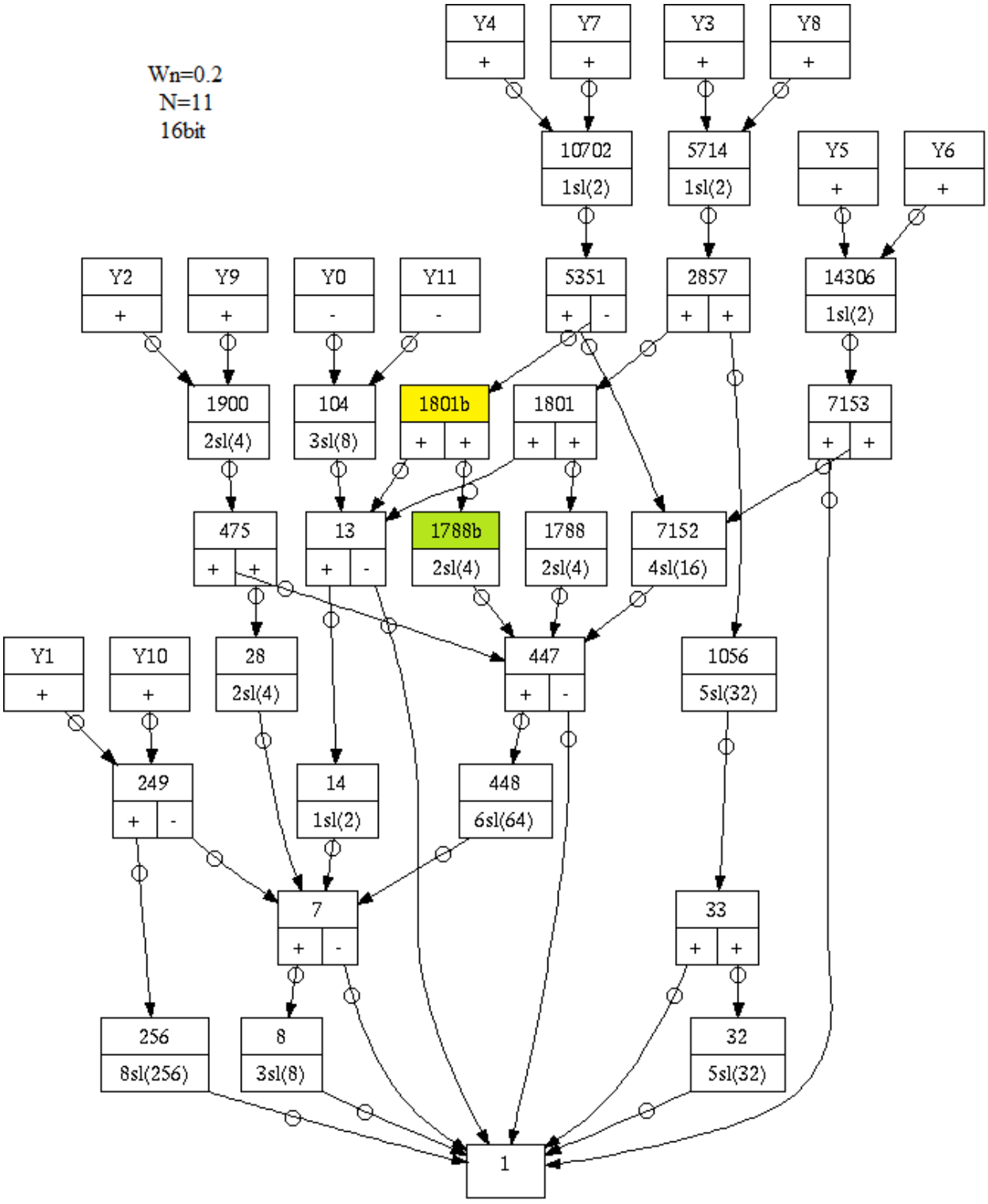


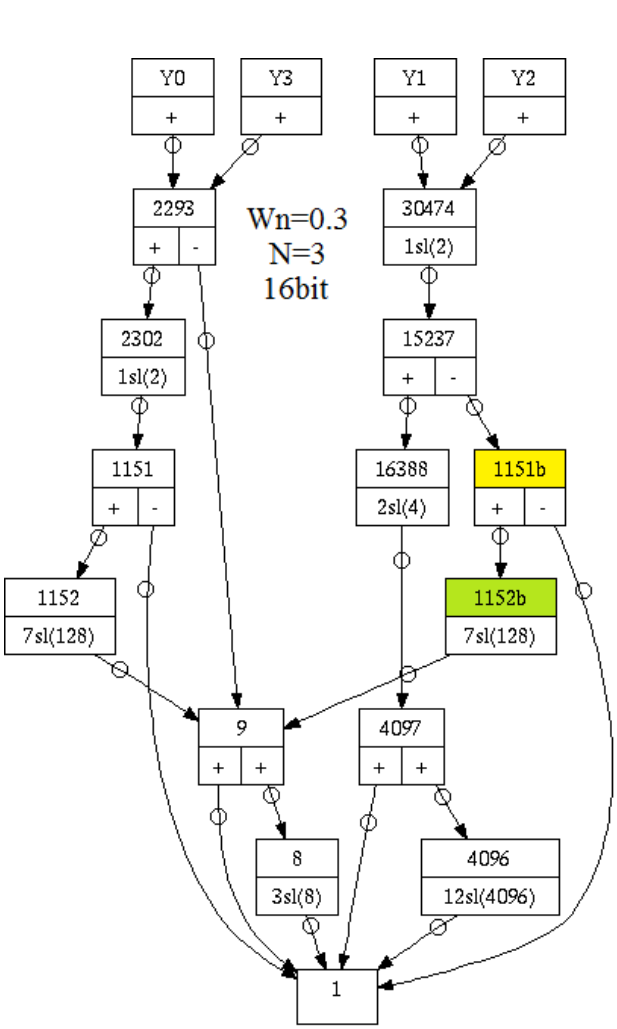

Y1
+
Y2
+
Wn=0.2
N=3
16bit
Y0
+
Y3
+
30041
+
-
2726
1sl(2)
2726b
1sl(2)
1363
+
-
1363b
+
-
1364
2sl(4)
1364b
2sl(4)
341
+
-
341b
+
-
352
5sl(32)
352b
5sl(32)
11
+
-
11b
+
-
5
+
+
32767
+
-
16b
4sl(16)
5b
+
+
16
4sl(16)
4
2sl(4)
32768
15sl(32768)
4b
2sl(4)
1

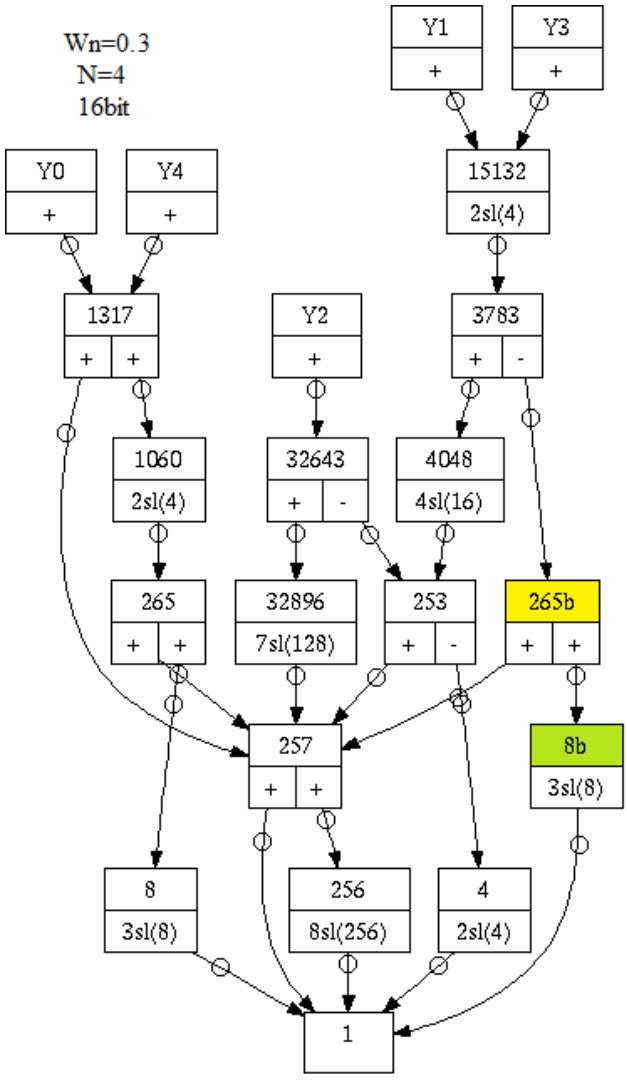

Wn=0.1
N=13
16bit
Y2
Y11
2018
1sl(2)
Y6
Y7
10171
1009
Y3
Y10
6108
2sl(4)
3054
1sl(2)
Y4
Y9
4063
1527
1527b
6553
Y1
Y12
Y5
Y8
1572
2sl(4)
1572b
2sl(4)
6288
4sl(16)
786
1sl(2)
8821
Y0
Y13
393
8428
2sl(4)
360
3sl(8)
265
2107
45
264
3sl(8)
2112
6sl(64)
40
3sl(8)
33
5
4096
12sl(4096)
4
2sl(4)
32
5sl(32)
1

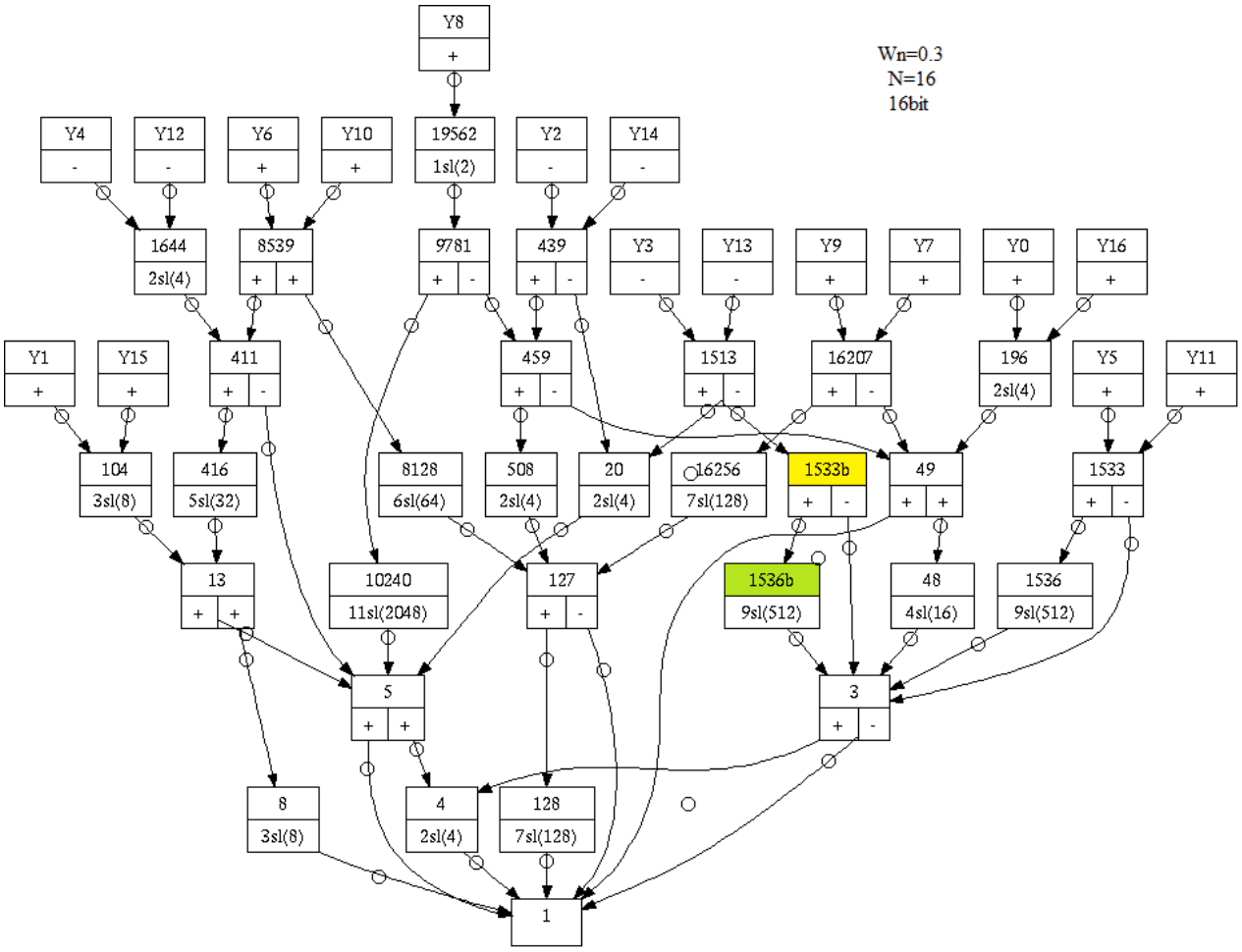

Wn=0.3
N=2
16bit
Y1
Y0
Y2
57619
3958
1sl(2)
7916
2sl(4)
1979
1979b
1984
6sl(64)
1984b
6sl(64)
31
5
65535
31b
5b
32
5sl(32)
4
2sl(4)
65536
16sl(65536)
32b
5sl(32)
4b
2sl(4)
1
Wn=0.2
N=4
16bit
Y1
Y0
Y2
57003
4266
1sl(2)
8532
2sl(4)
2133
2133b
85
85b
2048b
11sl(2048)
80
4sl(16)
2048
11sl(2048)
80b
4sl(16)
5
65535
5b
4
2sl(4)
65536
16sl(65536)
4b
2sl(4)
1

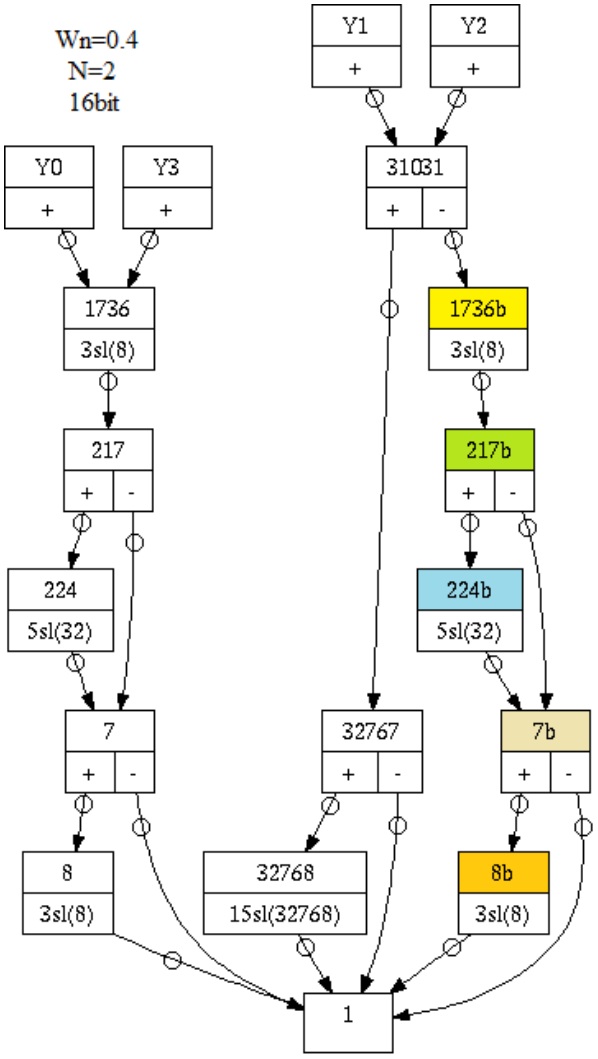

Wn=0.2
N=13
16bit
Y4
+
Y9
+
6219
+
-
Y5
+
Y8
+
Y3
+
Y10
+
9908
2sl(4)
10544
4sl(16)
Y13
-
Y0
-
Y12
-
Y1
-
2477
+
-
Y6
+
Y7
+
659
+
-
222
1sl(2)
163
+
+
Y2
+
Y11
+
13487
+
-
660
2sl(4)
111
+
+
158
1sl(2)
425
+
-
2640
4sl(16)
16127
+
-
165
+
+
79
+
-
165b
+
+
504
3sl(8)
16128
8sl(256)
160
5sl(32)
80
4sl(16)
160b
5sl(32)
63
+
-
5
+
+
64
6sl(64)
32
5sl(32)
4
2sl(4)
1

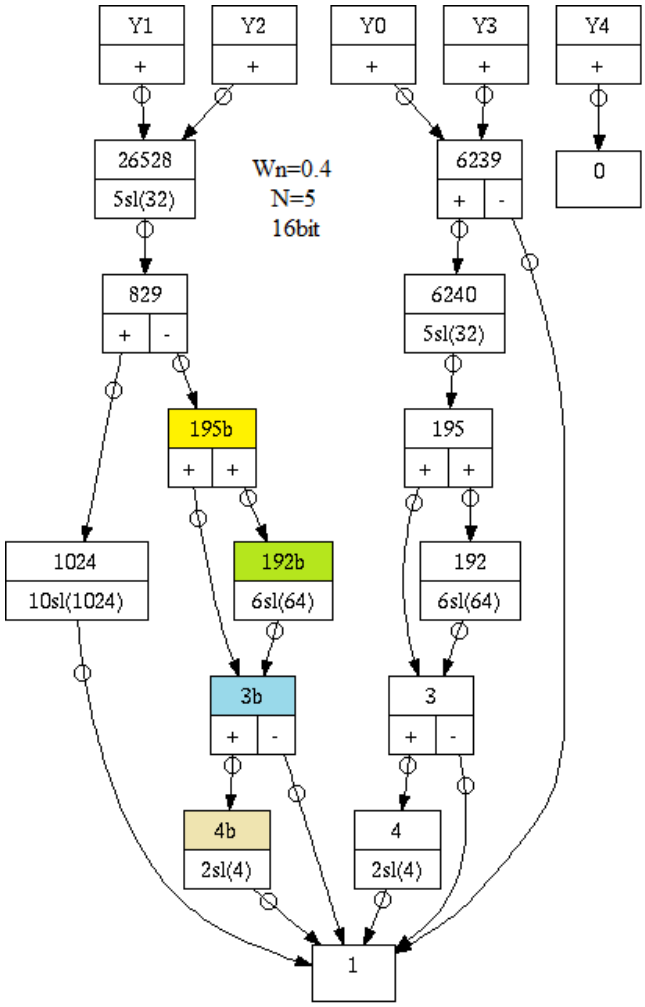

Wn=0.2
N=11
16bit
Y4
Y7
Y3
Y8
10702
1sl(2)
5714
1sl(2)
Y5
Y6
Y2
Y9
Y0
Y11
5351
2857
14306
1sl(2)
1900
2sl(4)
104
3sl(8)
1801b
1801
7153
475
13
1788b
2sl(4)
1788
2sl(4)
7152
4sl(16)
Y1
Y10
28
2sl(4)
447
1056
5sl(32)
249
14
1sl(2)
448
6sl(64)
7
33
256
8sl(256)
8
3sl(8)
32
5sl(32)
1

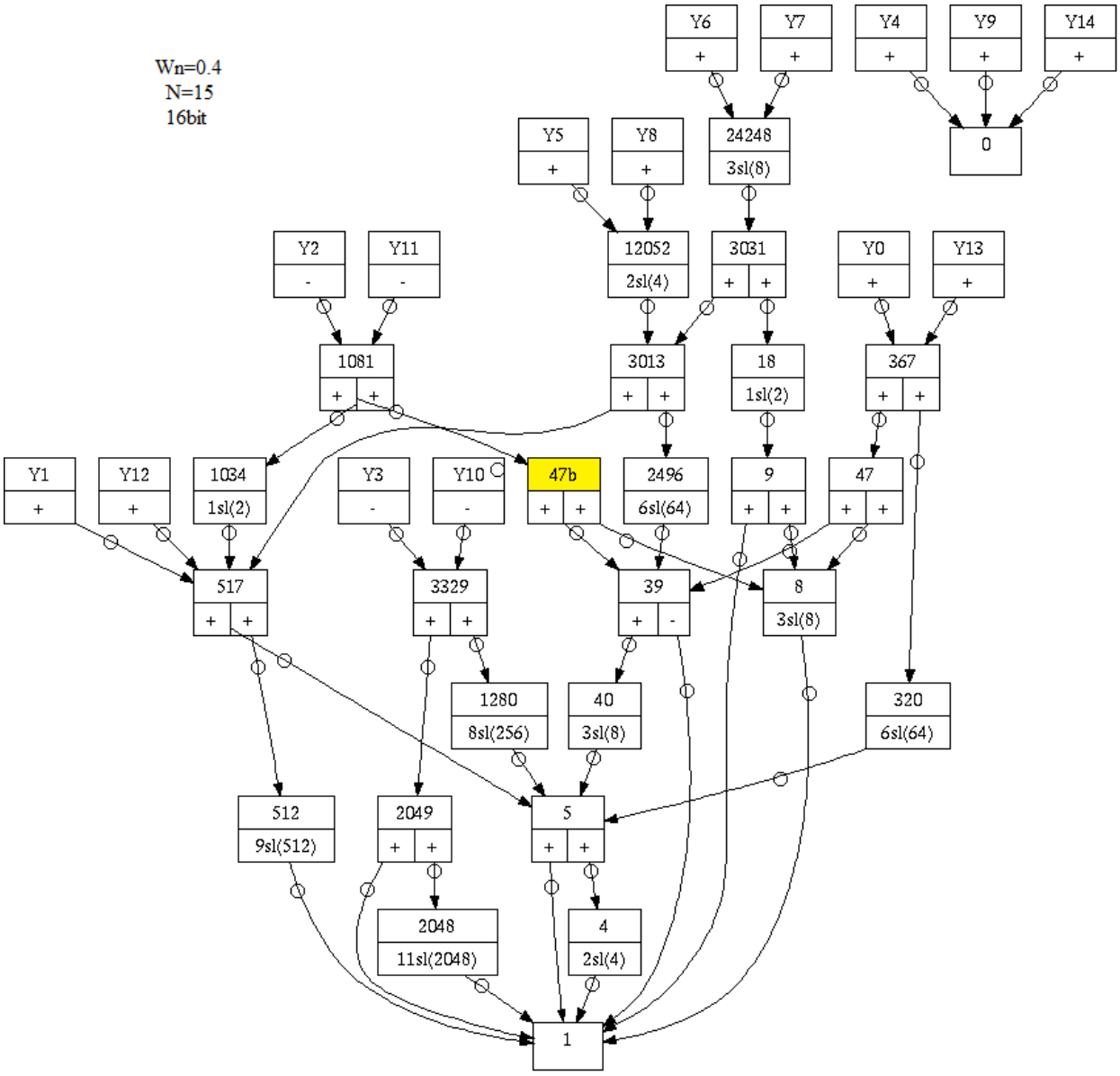

Y0
Y3
Y1
Y2
+
2293
+ -
30474
1sl(2)
Wn=0.3
N=3
16bit
2302
1sl(2)
15237
+ -
1151
+ -
16388
2sl(4)
1151b
+ -
1152
7sl(128)
1152b
7sl(128)
9
+ +
4097
+ +
8
3sl(8)
4096
12sl(4096)
1

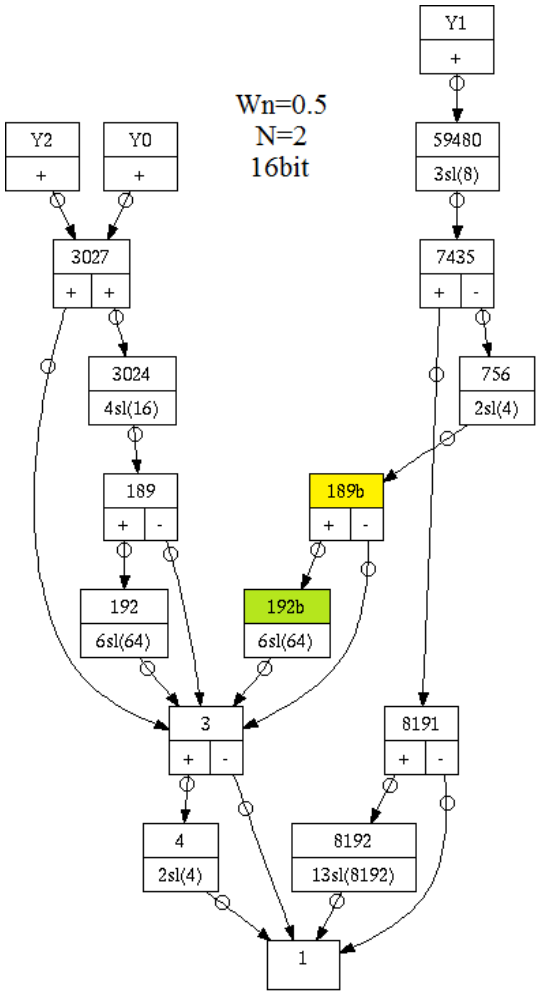

Wn=0.3
N=4
16bit
Y1
Y3
+
Y0
Y4
15132
2sl(4)
1317
+ +
Y2
3783
+ -
1060
2sl(4)
32643
+ -
4048
4sl(16)
265
+ +
32896
7sl(128)
253
+ -
265b
+ +
257
+ +
8b
3sl(8)
8
3sl(8)
256
8sl(256)
4
2sl(4)
1

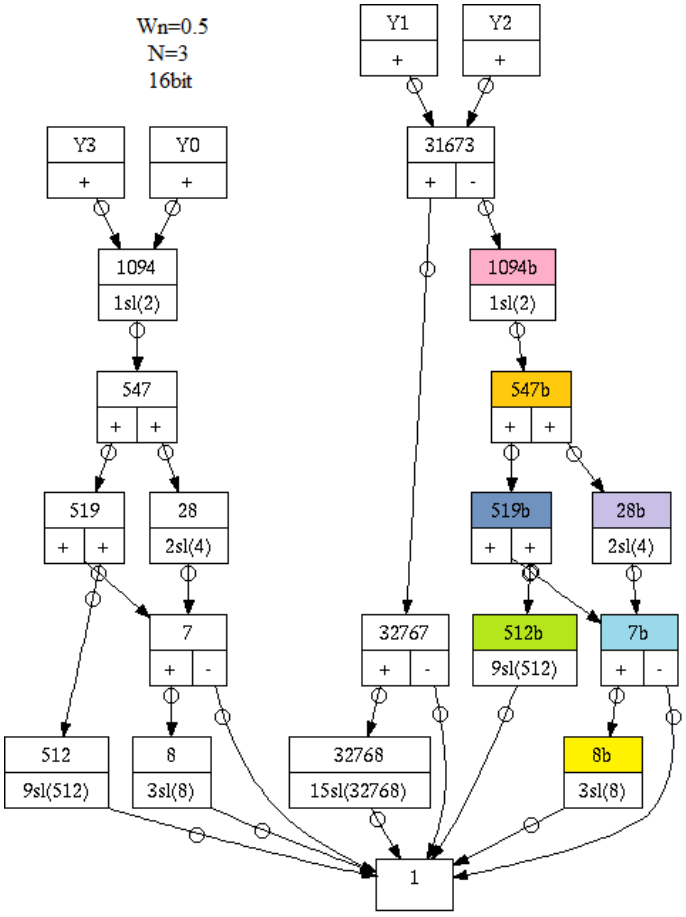

Wn=0.3
N=16
16bit
Y8
+
Y4
-
Y12
-
Y6
+
Y10
+
19562
1sl(2)
Y2
-
Y14
-
1644
2sl(4)
8539
+ +
9781
+ -
439
+ -
Y3
-
Y13
-
Y9
+
Y7
+
Y0
+
Y16
+
Y1
+
Y15
+
411
+ -
459
+ -
1513
+ -
16207
+ -
196
2sl(4)
Y5
+
Y11
+
104
3sl(8)
416
5sl(32)
8128
6sl(64)
508
2sl(4)
20
2sl(4)
16256
7sl(128)
1533b
+ -
49
+ +
1533
+ -
13
+ +
10240
11sl(2048)
127
+ -
1536b
9sl(512)
48
4sl(16)
1536
9sl(512)
5
+ +
3
+ -
8
3sl(8)
4
2sl(4)
128
7sl(128)
1

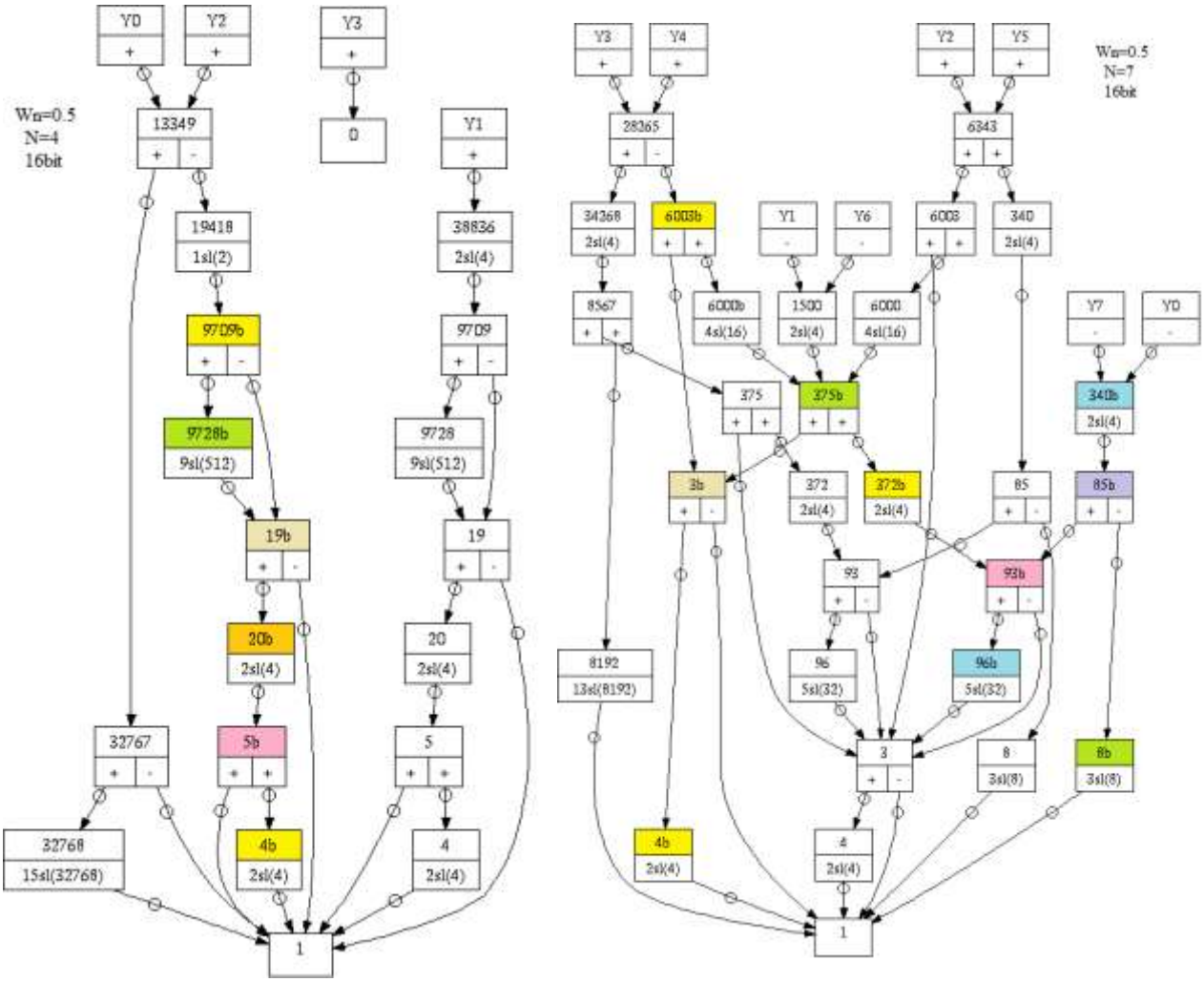

Wn=0.4
N=2
16bit
Y0
Y3
Y1
Y2
1736
3sl(8)
31031
1736b
3sl(8)
217
217b
224
5sl(32)
224b
5sl(32)
7
32767
7b
8
3sl(8)
32768
15sl(32768)
8b
3sl(8)
1

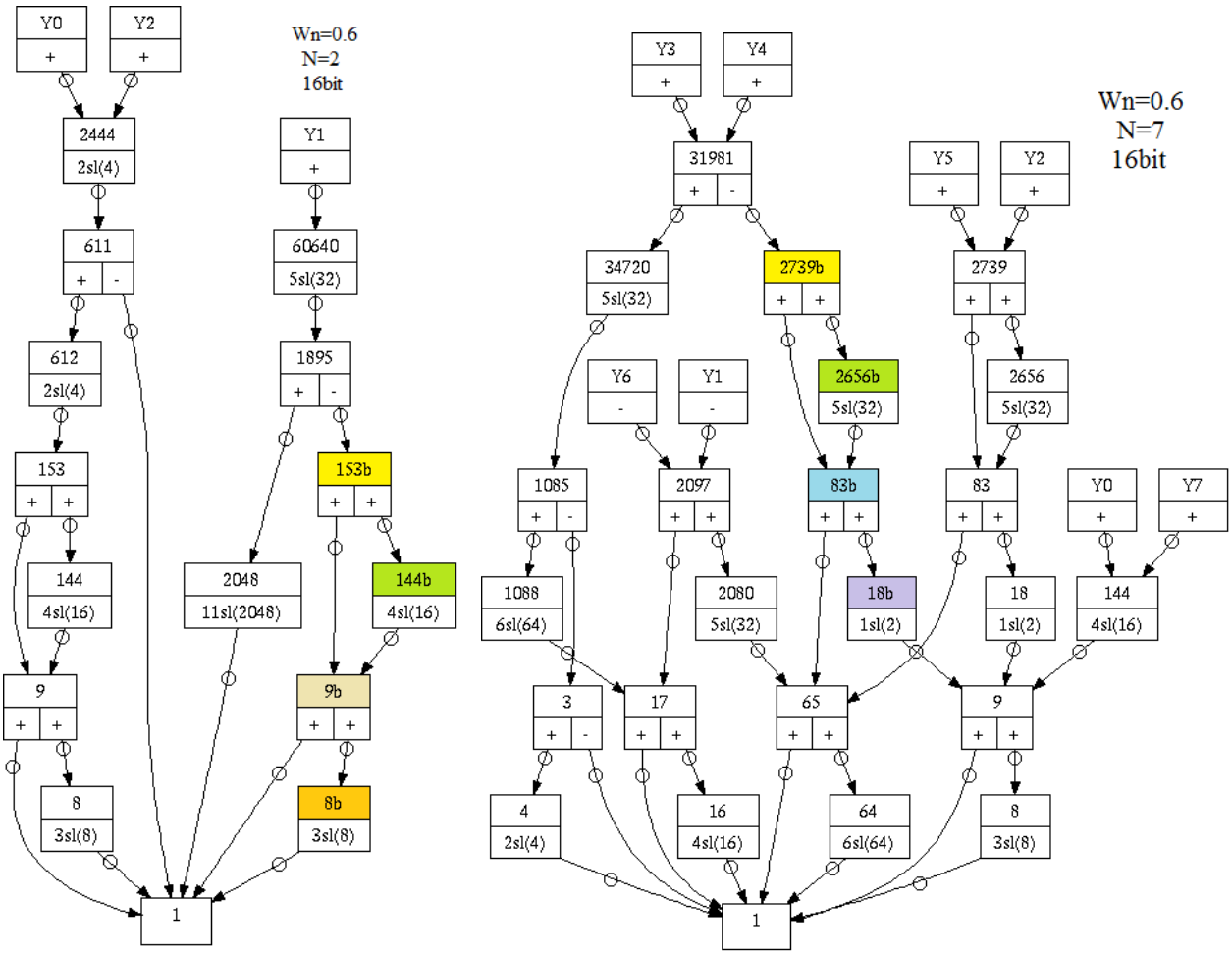

Y1
Y2
Y0
Y3
Y4
26528
5sl(32)
Wn=0.4
N=5
16bit
6239
0
829
6240
5sl(32)
195b
195
1024
10sl(1024)
192b
6sl(64)
192
6sl(64)
3b
3
4b
2sl(4)
4
2sl(4)
1

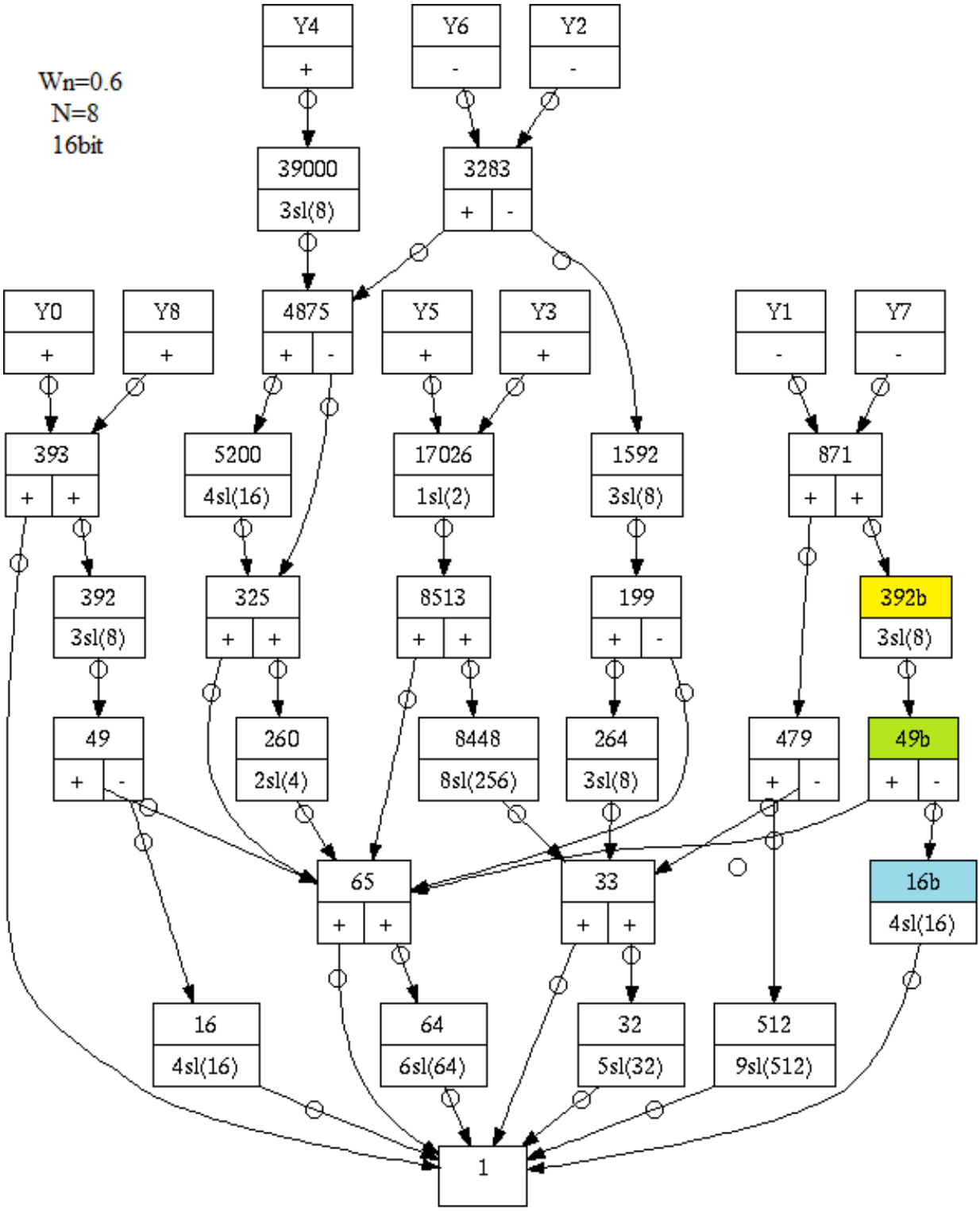

Wn=0.4
N=15
16bit
Y6
Y7
Y4
Y9
Y14
Y5
Y8
24248
3sl(8)
0
Y2
Y11
12052
2sl(4)
3031
Y0
Y13
1081
3013
18
1sl(2)
367
Y1
Y12
1034
1sl(2)
Y3
Y10
47b
2496
6sl(64)
9
47
517
3329
39
8
3sl(8)
1280
8sl(256)
40
3sl(8)
320
6sl(64)
512
9sl(512)
2049
5
2048
11sl(2048)
4
2sl(4)
1

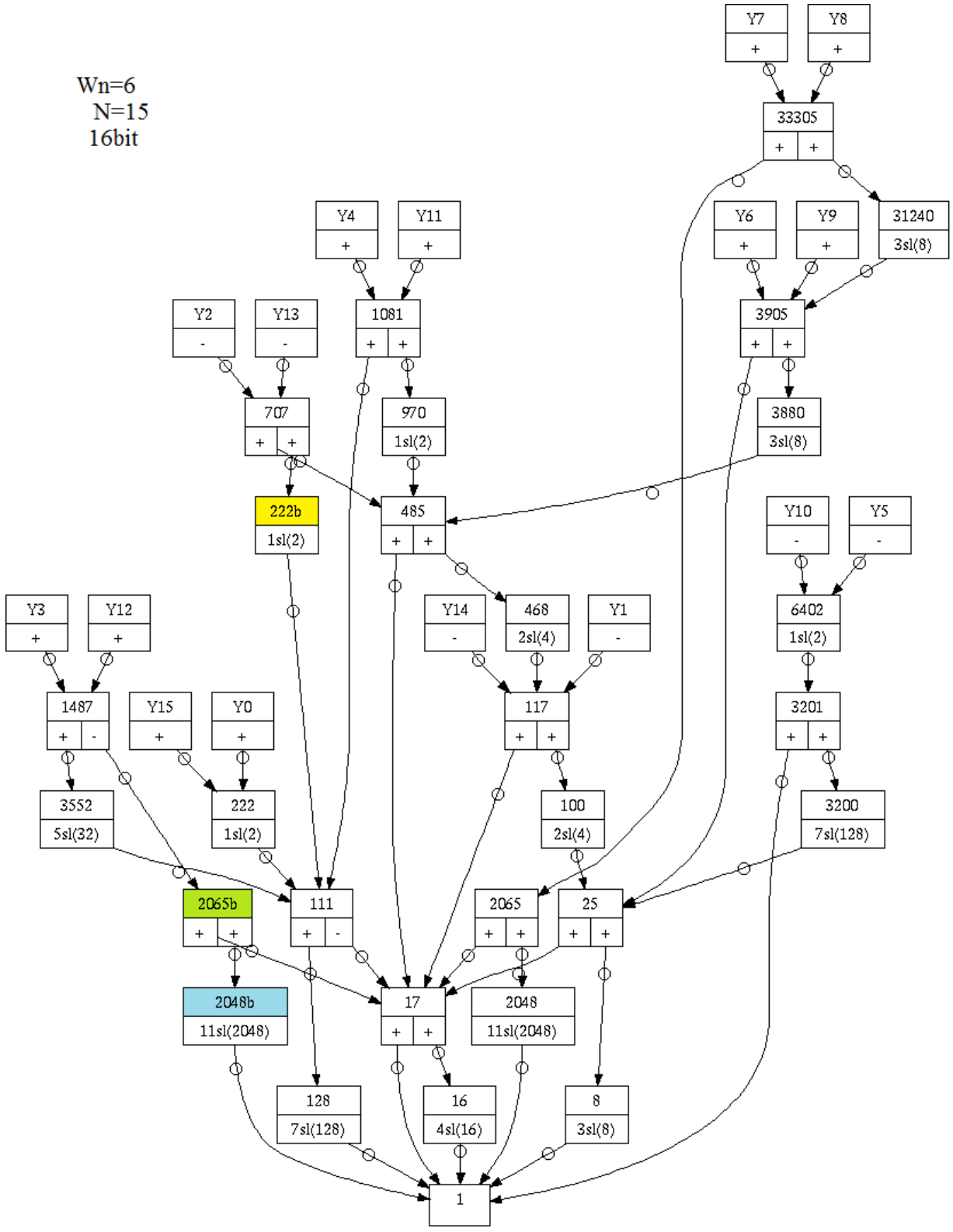

Wn=0.5
N=2
16bit
Y2
+
Y0
+
3027
+
+
3024
4sl(16)
189
+
-
192
6sl(64)
Y1
+
59480
3sl(8)
7435
+
-
756
2sl(4)
189b
+
-
192b
6sl(64)
3
+
-
4
2sl(4)
8191
+
-
8192
13sl(8192)
1

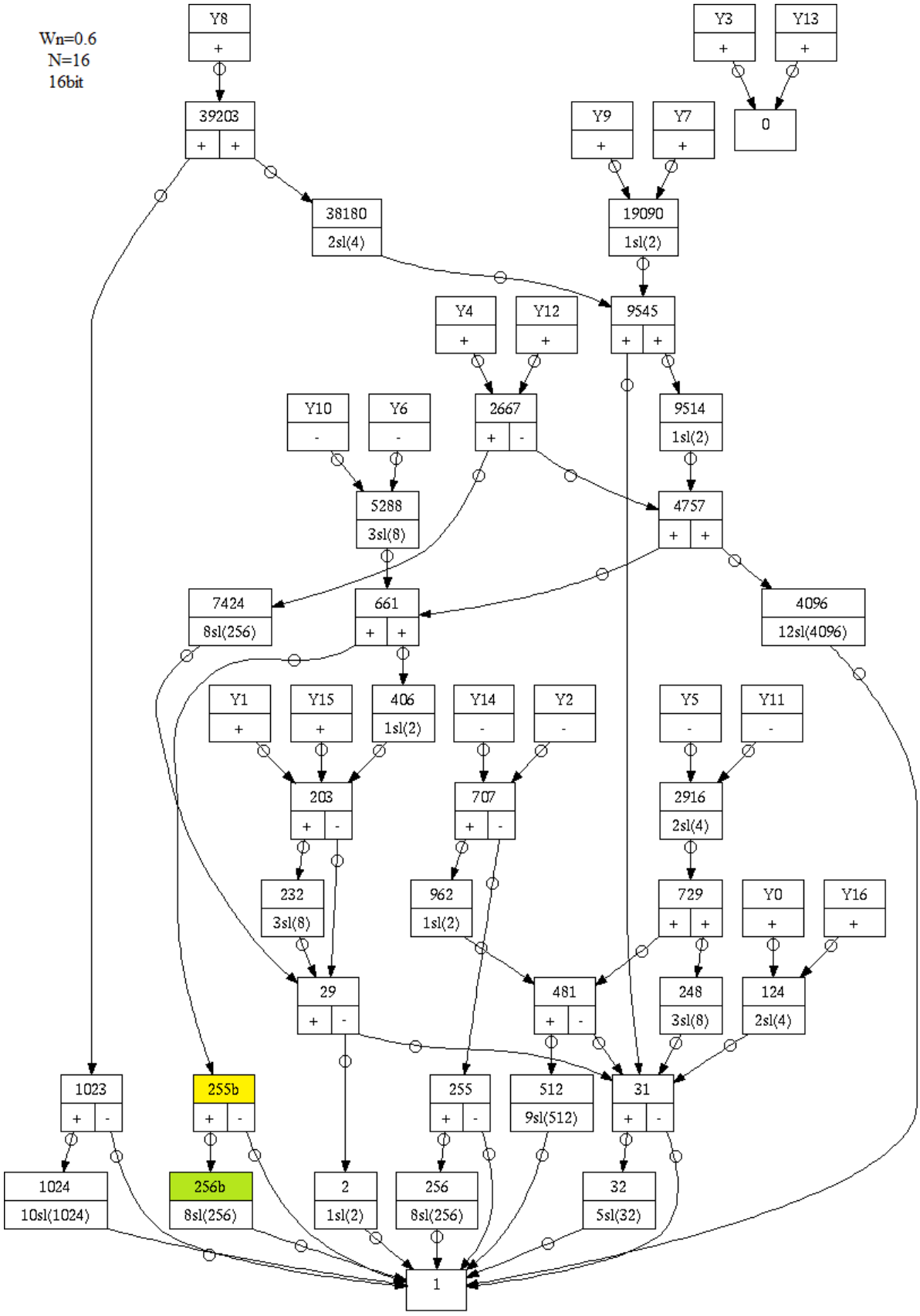

Wn=0.5
N=3
16bit
Y3
+
Y0
+
1094
1sl(2)
547
+
+
519
+
+
28
2sl(4)
7
+
-
512
9sl(512)
8
3sl(8)
Y1
+
Y2
+
31673
+
-
1094b
1sl(2)
547b
+
+
519b
+
+
28b
2sl(4)
32767
+
-
512b
9sl(512)
7b
+
-
32768
15sl(32768)
8b
3sl(8)
1

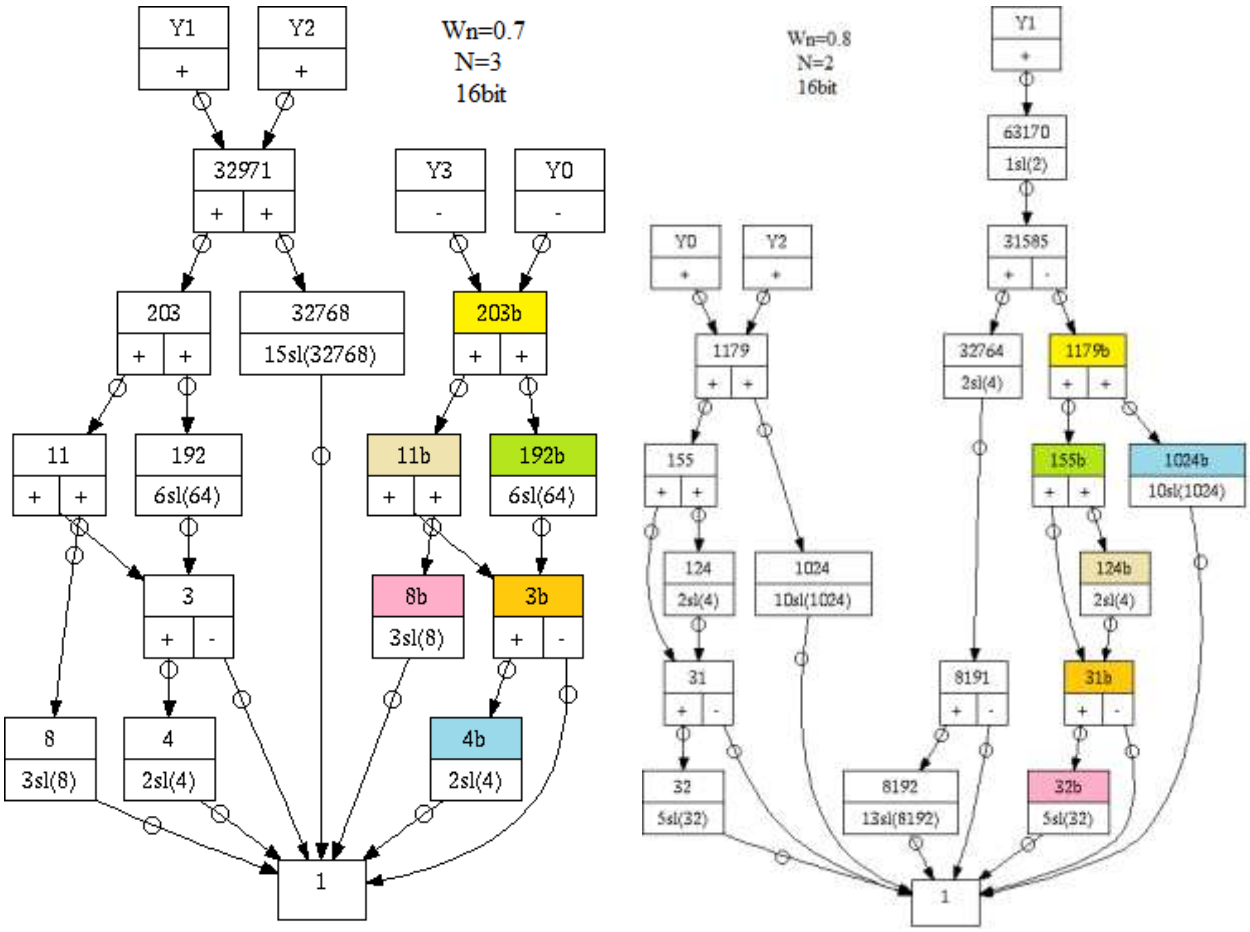

Wn=0.5
N=4
16bit
Y0
Y2
Y3
13349
0
Y1
19418
1sl(2)
38836
2sl(4)
9709b
9709
9728b
9sl(512)
9728
9sl(512)
19b
19
20b
2sl(4)
20
2sl(4)
32767
5b
5
32768
15sl(32768)
4b
2sl(4)
4
2sl(4)
1
Wn=0.5
N=7
16bit
Y3
Y4
Y2
Y5
28365
6343
34368
2sl(4)
6003b
Y1
Y6
6003
340
2sl(4)
8567
6000b
4sl(16)
1500
2sl(4)
6000
4sl(16)
Y7
Y0
375
375b
340b
2sl(4)
3b
372
2sl(4)
372b
2sl(4)
85
85b
93
93b
8192
13sl(8192)
96
5sl(32)
96b
5sl(32)
3
8
3sl(8)
8b
3sl(8)
4b
2sl(4)
4
2sl(4)
1

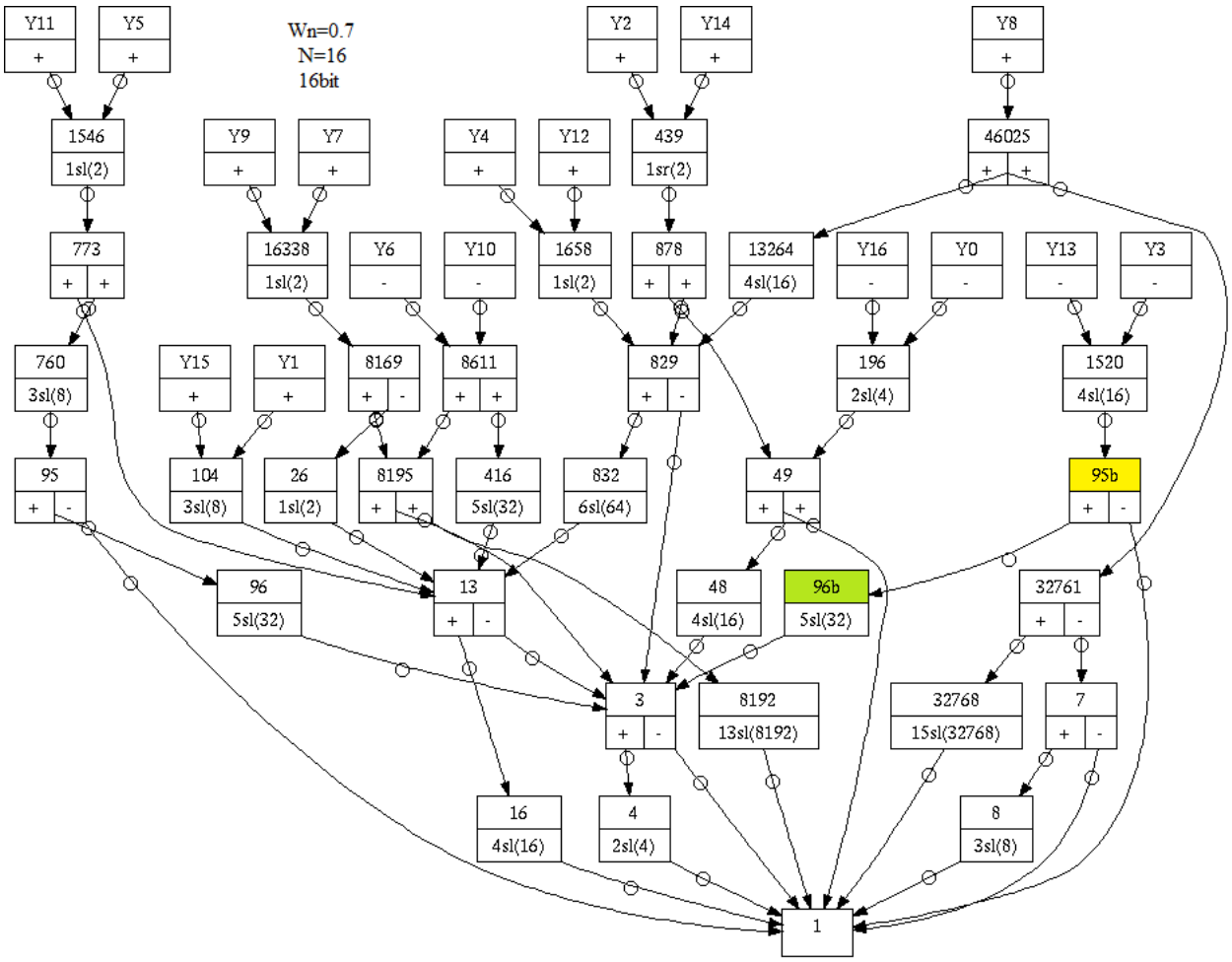

Wn=0.6
N=2
16bit
Y0
+
Y2
+
2444
2sl(4)
611
+ -
612
2sl(4)
153
+ +
144
4sl(16)
9
+ +
8
3sl(8)
1
Y1
+
60640
5sl(32)
1895
+ -
153b
+ +
2048
11sl(2048)
144b
4sl(16)
9b
+ +
8b
3sl(8)
Wn=0.6
N=7
16bit
Y3
+
Y4
+
31981
+ -
34720
5sl(32)
2739b
+ +
Y6
-
Y1
-
2656b
5sl(32)
1085
+ -
2097
+ +
83b
+ +
1088
6sl(64)
2080
5sl(32)
18b
1sl(2)
3
+ -
17
+ +
65
+ +
4
2sl(4)
16
4sl(16)
64
6sl(64)
1
Y5
+
Y2
+
2739
+ +
2656
5sl(32)
83
+ +
Y0
+
Y7
+
18
1sl(2)
144
4sl(16)
9
+ +
8
3sl(8)

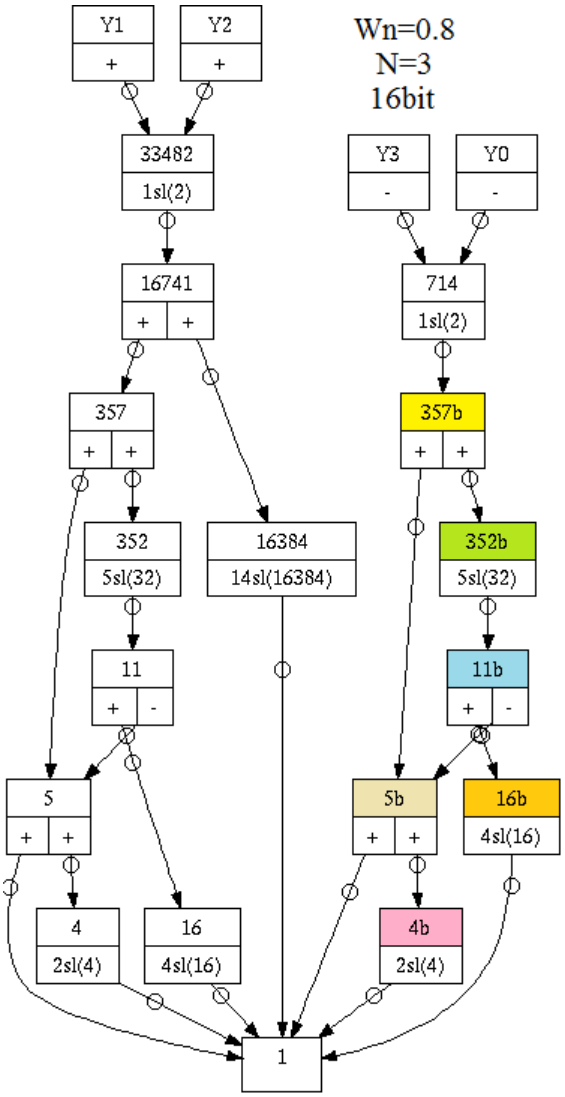

Wn=0.6
N=8
16bit
Y4
+
Y6
-
Y2
-
39000
3sl(8)
3283
+
-
Y0
+
Y8
+
4875
+
-
Y5
+
Y3
+
Y1
-
Y7
-
393
+
+
5200
4sl(16)
17026
1sl(2)
1592
3sl(8)
871
+
+
392
3sl(8)
325
+
+
8513
+
+
199
+
-
392b
3sl(8)
49
+
-
260
2sl(4)
8448
8sl(256)
264
3sl(8)
479
+
-
49b
+
-
65
+
+
33
+
+
16b
4sl(16)
16
4sl(16)
64
6sl(64)
32
5sl(32)
512
9sl(512)
1

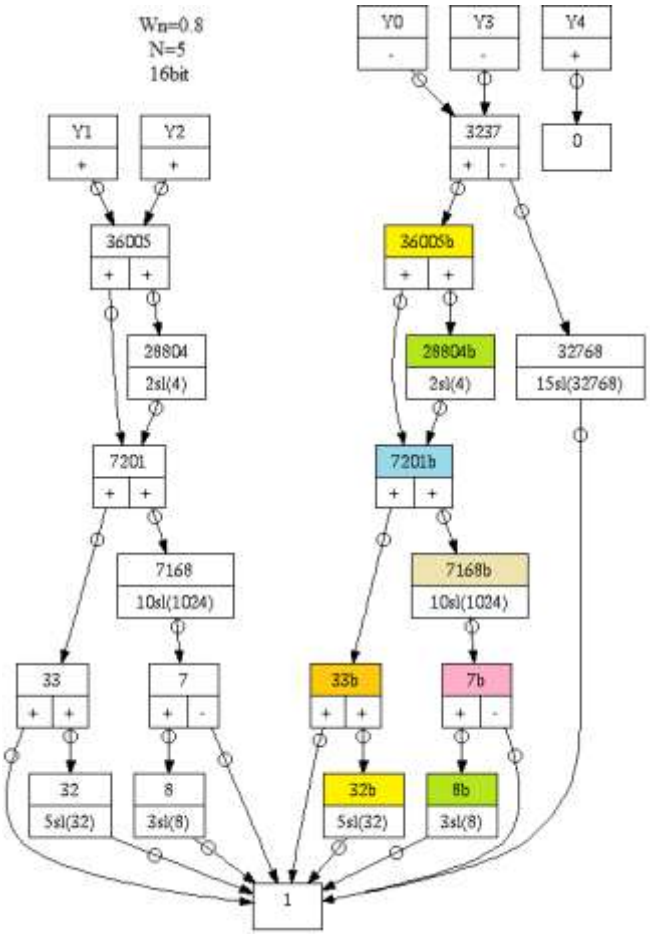

Wn=6
N=15
16bit
Y7
+
Y8
+
33305
+
+
Y4
+
Y11
+
Y6
+
Y9
+
31240
3sl(8)
Y2
-
Y13
-
1081
+
+
3905
+
+
707
+
+
970
1sl(2)
3880
3sl(8)
222b
1sl(2)
485
+
+
Y10
-
Y5
-
Y3
+
Y12
+
Y14
-
468
2sl(4)
Y1
-
6402
1sl(2)
1487
+
-
Y15
+
Y0
+
117
+
+
3201
+
+
3552
5sl(32)
222
1sl(2)
100
2sl(4)
3200
7sl(128)
2065b
+
+
111
+
-
2065
+
+
25
+
+
2048b
11sl(2048)
17
+
+
2048
11sl(2048)
128
7sl(128)
16
4sl(16)
8
3sl(8)
1

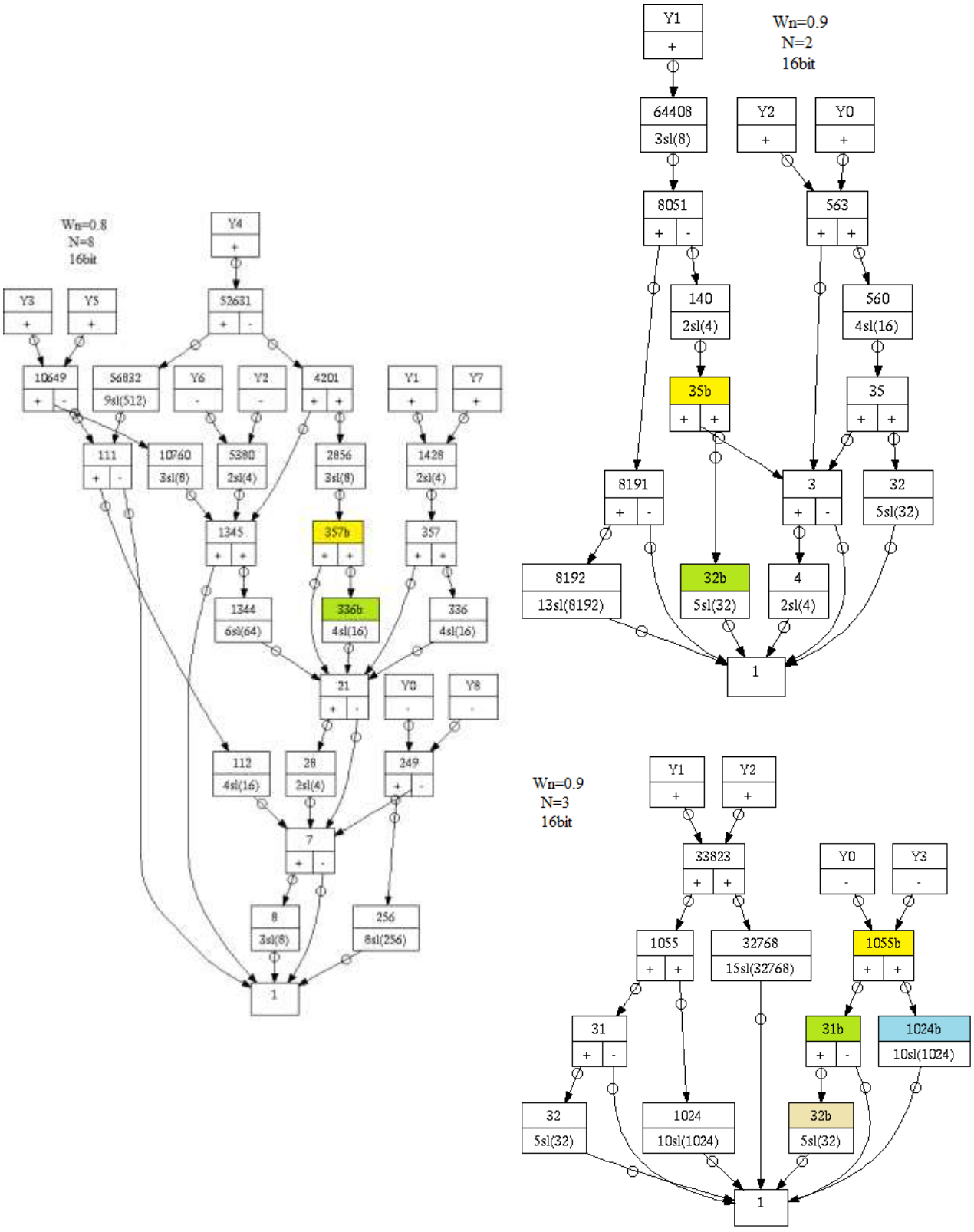

Wn=0.6
N=16
16bit
Y8
+
39203
+ +
38180
2sl(4)
Y3
+
Y13
+
0
Y9
+
Y7
+
19090
1sl(2)
Y4
+
Y12
+
9545
+ +
Y10
-
Y6
-
2667
+ -
9514
1sl(2)
5288
3sl(8)
4757
+ +
7424
8sl(256)
661
+ +
4096
12sl(4096)
Y1
+
Y15
+
406
1sl(2)
Y14
-
Y2
-
Y5
-
Y11
-
203
+ -
707
+ -
2916
2sl(4)
232
3sl(8)
962
1sl(2)
729
+ +
Y0
+
Y16
+
29
+ -
481
+ -
248
3sl(8)
124
2sl(4)
1023
+ -
255b
+ -
255
+ -
512
9sl(512)
31
+ -
1024
10sl(1024)
256b
8sl(256)
2
1sl(2)
256
8sl(256)
32
5sl(32)
1

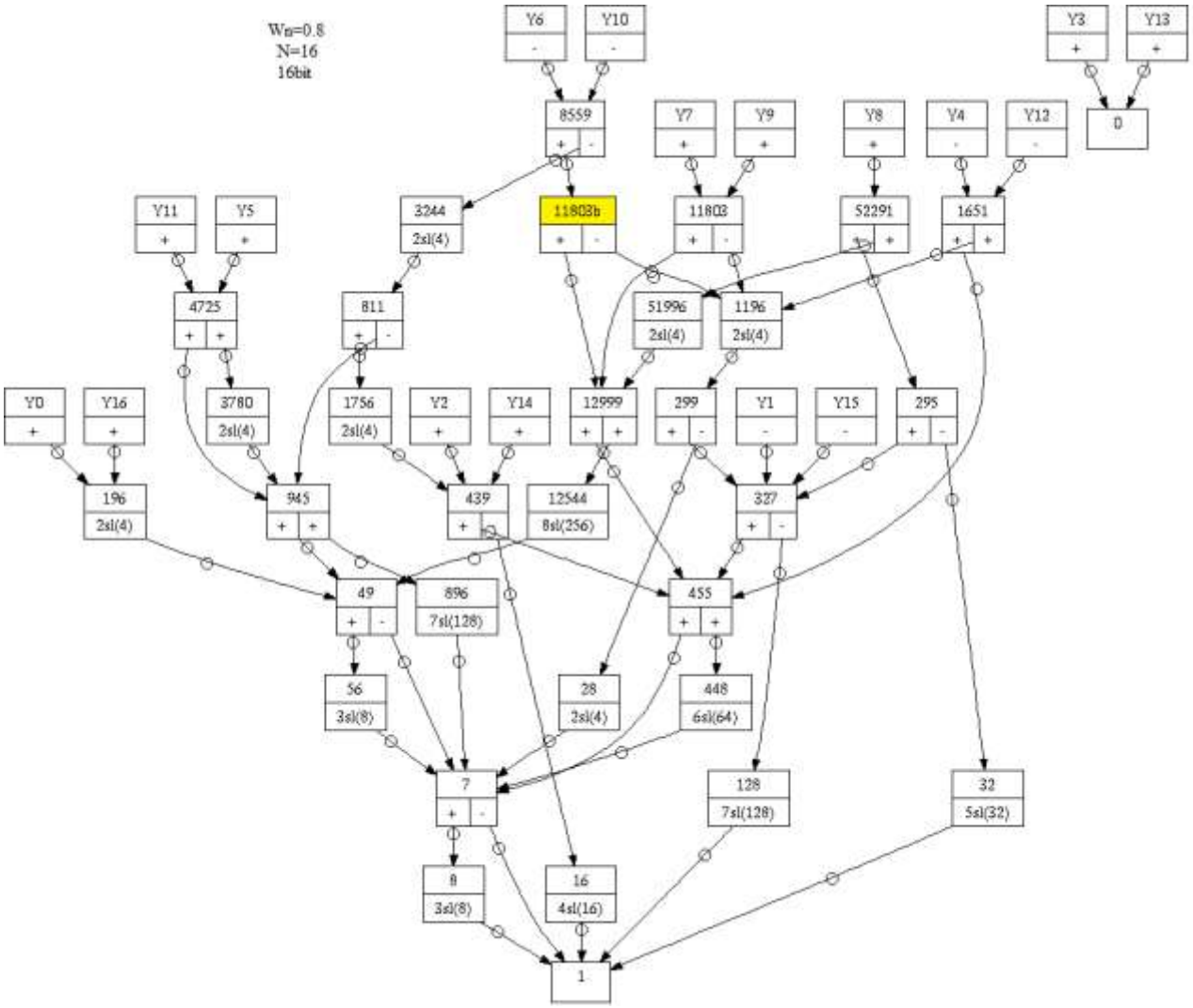

Wn=0.7
N=3
16bit
Y1
Y2
32971
Y3
Y0
203
32768
15sl(32768)
203b
11
192
6sl(64)
11b
192b
6sl(64)
3
8b
3sl(8)
3b
8
3sl(8)
4
2sl(4)
4b
2sl(4)
1
Wn=0.8
N=2
16bit
Y1
63170
1sl(2)
Y0
Y2
31585
1179
32764
2sl(4)
1179b
155
155b
1024b
10sl(1024)
124
2sl(4)
1024
10sl(1024)
124b
2sl(4)
31
8191
31b
32
5sl(32)
8192
13sl(8192)
32b
5sl(32)
1

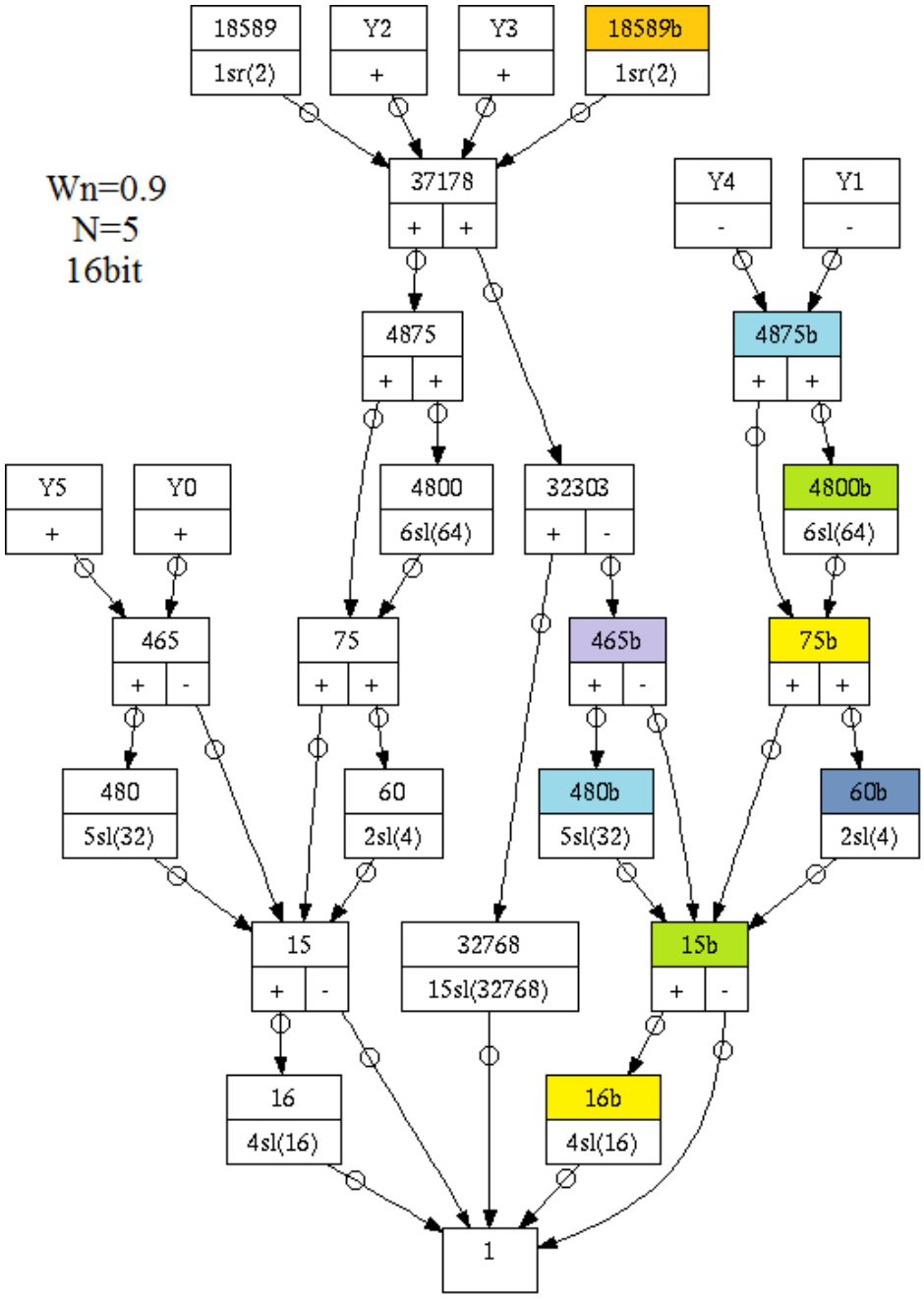

Wn=0.7
N=16
16bit
Y11
Y5
Y2
Y14
Y8
1546
1sl(2)
Y9
Y7
Y4
Y12
439
1sr(2)
46025
773
16338
1sl(2)
Y6
Y10
1658
1sl(2)
878
13264
4sl(16)
Y16
Y0
Y13
Y3
760
3sl(8)
Y15
Y1
8169
8611
829
196
2sl(4)
1520
4sl(16)
95
104
3sl(8)
26
1sl(2)
8195
416
5sl(32)
832
6sl(64)
49
95b
96
5sl(32)
13
48
4sl(16)
96b
5sl(32)
32761
3
8192
13sl(8192)
32768
15sl(32768)
7
16
4sl(16)
4
2sl(4)
8
3sl(8)
1

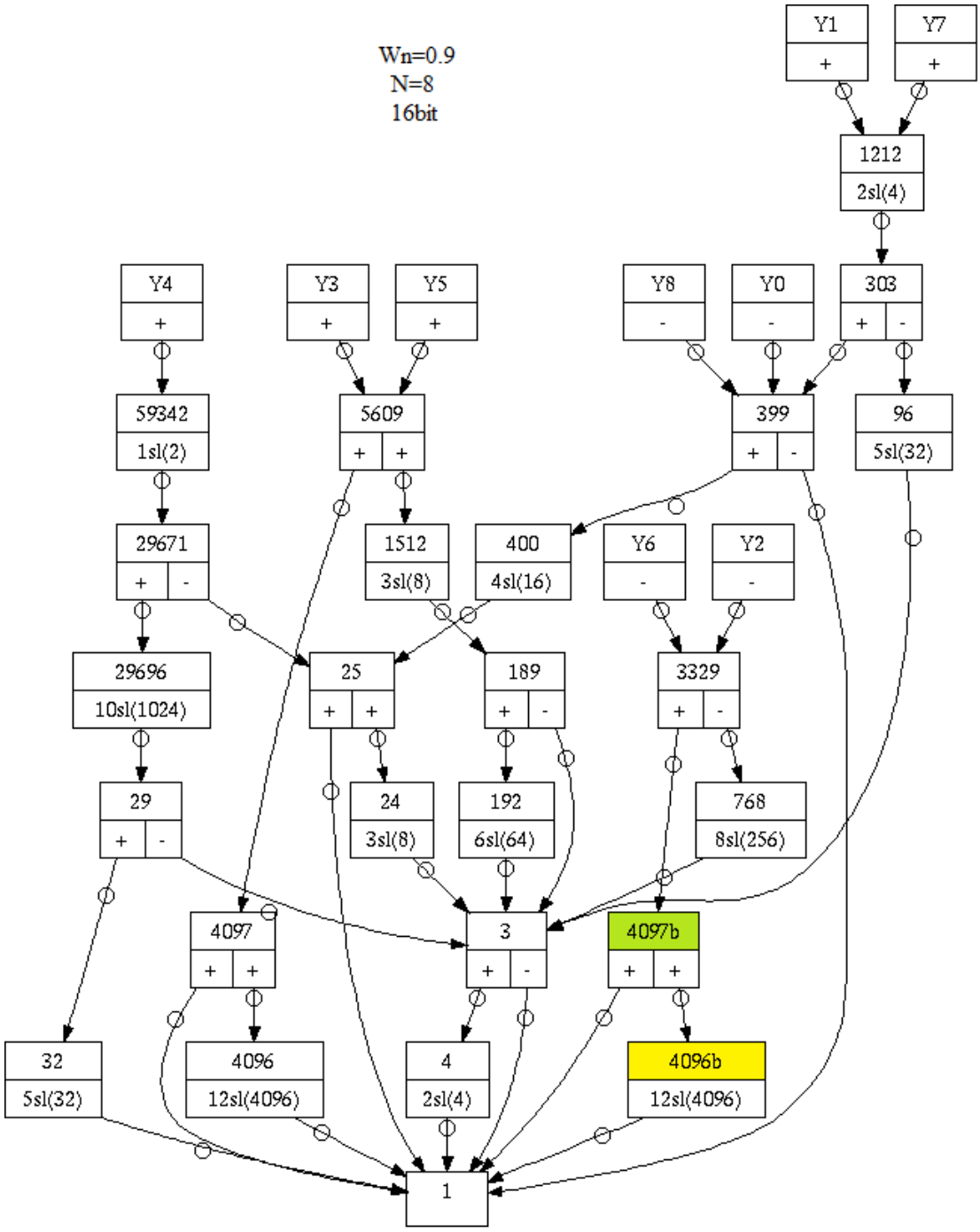

Wn=0.8
N=3
16bit
Y1
Y2
33482
1sl(2)
16741
357
352
5sl(32)
11
5
4
2sl(4)
16
4sl(16)
16384
14sl(16384)
Y3
Y0
714
1sl(2)
357b
352b
5sl(32)
11b
5b
4b
2sl(4)
16b
4sl(16)
1

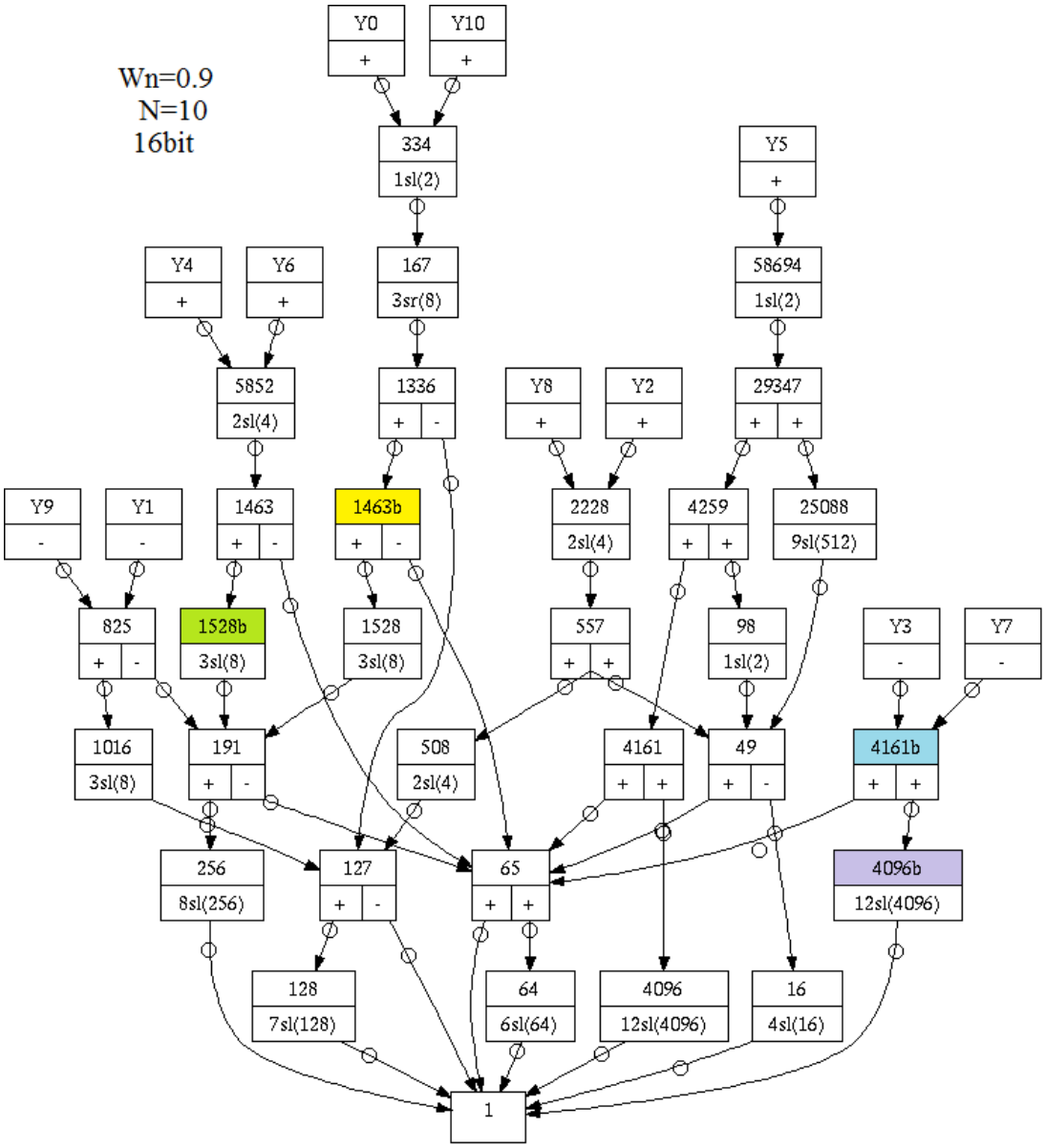

Wn=0.8
N=5
16bit
Y1
Y2
36005
28804
2sl(4)
7201
7168
10sl(1024)
33
32
5sl(32)
7
8
3sl(8)
Y0
Y3
Y4
3237
0
36005b
28804b
2sl(4)
32768
15sl(32768)
7201b
7168b
10sl(1024)
33b
32b
5sl(32)
7b
8b
3sl(8)
1

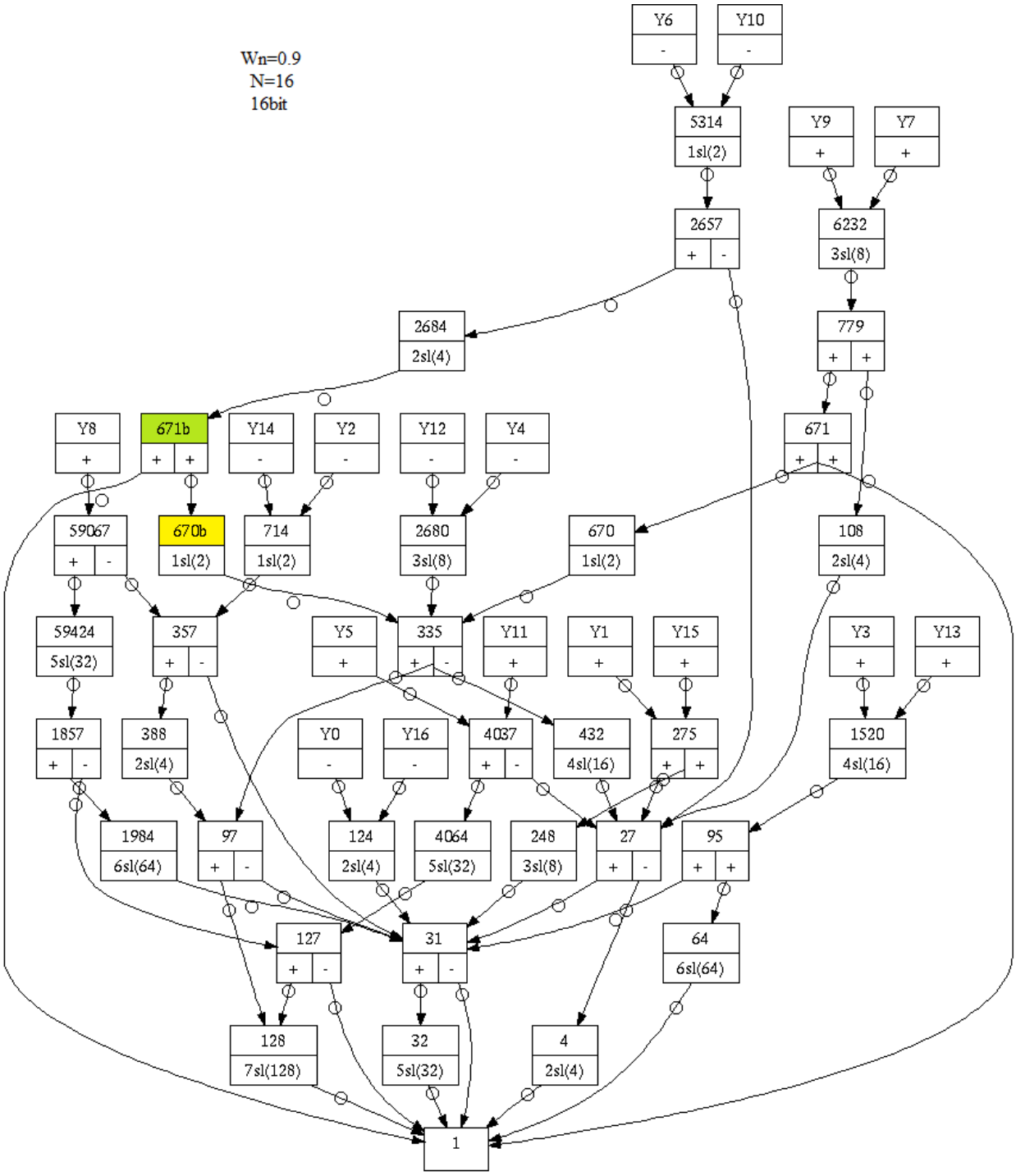

Wn=0.8
N=8
16bit
Y4
Y3
Y5
52631
10649
56832
9sl(512)
Y6
Y2
4201
Y1
Y7
111
10760
3sl(8)
5380
2sl(4)
2856
3sl(8)
1428
2sl(4)
1345
357b
357
1344
6sl(64)
336b
4sl(16)
336
4sl(16)
21
Y0
Y8
112
4sl(16)
28
2sl(4)
249
7
8
3sl(8)
256
8sl(256)
1
Wn=0.9
N=2
16bit
Y1
64408
3sl(8)
Y2
Y0
8051
563
140
2sl(4)
560
4sl(16)
35b
35
8191
3
32
5sl(32)
8192
13sl(8192)
32b
5sl(32)
4
2sl(4)
1
Wn=0.9
N=3
16bit
Y1
Y2
33823
Y0
Y3
1055
32768
15sl(32768)
1055b
31
31b
1024b
10sl(1024)
32
5sl(32)
1024
10sl(1024)
32b
5sl(32)
1


Wn=0.8
N=16
16bit
Y6
Y10
Y3
Y13
8559
Y7
Y9
Y8
Y4
Y12
0
Y11
Y5
3244
2sl(4)
11803h
11803
52291
1651
4725
811
51996
2sl(4)
1196
2sl(4)
Y0
Y16
3780
2sl(4)
1756
2sl(4)
Y2
Y14
12999
299
Y1
Y15
295
196
2sl(4)
945
439
12544
8sl(256)
327
49
896
7sl(128)
455
56
3sl(8)
28
2sl(4)
448
6sl(64)
7
128
7sl(128)
32
5sl(32)
9
3sl(8)
16
4sl(16)
1


Wn=0.9
N=5
16bit
18589
1sr(2)
Y2
+
Y3
+
18589b
1sr(2)
37178
+ +
Y4
-
Y1
-
4875
+ +
4875b
+ +
Y5
+
Y0
+
4800
6sl(64)
32303
+ -
4800b
6sl(64)
465
+ -
75
+ +
465b
+ -
75b
+ +
480
5sl(32)
60
2sl(4)
480b
5sl(32)
60b
2sl(4)
15
+ -
32768
15sl(32768)
15b
+ -
16
4sl(16)
16b
4sl(16)
1


Wn=0.9
N=8
16bit
Y1
+
Y7
+
1212
2sl(4)
Y4
+
Y3
+
Y5
+
Y8
-
Y0
-
303
+
-
59342
1sl(2)
5609
+
+
399
+
-
96
5sl(32)
29671
+
-
1512
3sl(8)
400
4sl(16)
Y6
-
Y2
-
29696
10sl(1024)
25
+
+
189
+
-
3329
+
-
29
+
-
24
3sl(8)
192
6sl(64)
768
8sl(256)
4097
+
+
3
+
-
4097b
+
+
32
5sl(32)
4096
12sl(4096)
4
2sl(4)
4096b
12sl(4096)
1


Wn=0.9
N=10
16bit
Y0
Y10
334
1sl(2)
Y5
58694
1sl(2)
Y4
Y6
167
3sr(8)
5852
2sl(4)
1336
Y8
Y2
29347
Y9
Y1
1463
1463b
2228
2sl(4)
4259
25088
9sl(512)
825
1528b
3sl(8)
1528
3sl(8)
557
98
1sl(2)
Y3
Y7
1016
3sl(8)
191
508
2sl(4)
4161
49
4161b
256
8sl(256)
127
65
4096b
12sl(4096)
128
7sl(128)
64
6sl(64)
4096
12sl(4096)
16
4sl(16)
1


Wn=0.9
N=16
16bit
Y6
-
Y10
-
5314
1sl(2)
Y9
+
Y7
+
2657
+ -
6232
3sl(8)
2684
2sl(4)
779
+ +
Y8
+
671b
+ +
Y14
-
Y2
-
Y12
-
Y4
-
671
+ +
59067
+ -
670b
1sl(2)
714
1sl(2)
2680
3sl(8)
670
1sl(2)
108
2sl(4)
59424
5sl(32)
357
+ -
Y5
+
335
+ -
Y11
+
Y1
+
Y15
+
Y3
+
Y13
+
1857
+ -
388
2sl(4)
Y0
-
Y16
-
4037
+ -
432
4sl(16)
275
+ +
1520
4sl(16)
1984
6sl(64)
97
+ -
124
2sl(4)
4064
5sl(32)
248
3sl(8)
27
+ -
95
+ +
127
+ -
31
+ -
64
6sl(64)
128
7sl(128)
32
5sl(32)
4
2sl(4)
1

Anexo A-8 Codigo fuente del Script de inserción de errores.

```
# Parámetros iniciales del script
add wave -position insertpoint  \
sim:/acm_filter_MultiplyBlock/clk

force -freeze sim:/acm_filter_MultiplyBlock/clk 1 0, 0 {50 ns} -r 100

# Lectura de errores del fichero DetectError
set infile [open "C:/Modeltech_pe_edu_10.1b/error.dat" r]
set m sim:/acm_filter_MultiplyBlock/w1
while { [gets $infile line] >= 0 } {

        force X $line ;
        run 100;
        set r sim:/acm_filter_MultiplyBlock/

        if { [gets $infile line] > 0 } {

                set m [concat $r$line]

                set x [ examine $m ]

                if { $x == "St1"}  {
                force -freeze $m St0 0;
                } else {
                force -freeze $m St1 0;
                }
        }
        run 100;
        noforce $m;


}
# Cierre del fichero de errores.
close $infile
```

Anexo A-9 Tablas.

En este anexo se recogen la práctica totalidad de las tablas correspondientes al capítulo 4 de esta Memoria, con objeto de hacer más cómoda la lectura de dicho capítulo. Se presentan en este Anexo con el mismo número que se indica en el texto del que proceden.

Tabla 4.1 Área del Bloque Multiplicador Filtros FIR Paso bajo de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	14316	16502	17170	14097	10624	12450	15993	13278	9417
15	12560	12780	18271	12357	16435	10561	11389	8445	11678
14	16432	10019	12357	14822	8698	15387	10255	12906	11845
13	10275	12710	11436	12267	12547	10228	10115	9207	14020
12	18887	15048	15348	13841	11043	12217	12434	9802	14483
11	12803	12876	9287	10877	9999	13611	12520	10401	11396
10	12740	12574	12893	12288	11227	12358	12640	9630	11612
9	10182	10375	12667	8848	13033	9118	10874	9650	9204
8	13395	11080	11150	9953	8818	12817	13049	6799	9417
7	11366	8795	8163	6041	12740	8679	9647	9650	9650
6	11044	6180	11107	12710	6041	6041	7225	9417	9650
5	8862	11316	9813	8449	6320	7222	7454	9274	9204
4	8662	6180	10937	9044	9770	6799	7012	4817	9700
3	3632	3632	0	2408	4604	7225	4883	7079	7012
2	4604	4604	7225	6799	7012	7531	5123	5046	7548

Tabla 4.2 Área total para implementación de Filtros FIR Paso bajo de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	86672	84467	82687	75223	69807	73576	81510	81243	81772
15	66594	69262	85984	73230	84147	73882	72262	76157	79391
14	79500	59409	68587	73500	56146	74065	66484	71584	81752
13	55021	66744	63021	70692	70972	64262	68540	60793	79284
12	79507	61990	69129	70070	56043	68447	66215	56745	75103
11	55101	64462	58424	60014	59136	58358	68497	52700	67372
10	64073	57068	64226	56782	53778	56852	63973	54124	62945
9	50032	52674	52517	55538	59722	44577	53173	49500	55894
8	55441	46287	46357	51998	39634	54862	55095	48845	51463
7	41929	41806	45565	36604	50142	41690	40210	47052	47052
6	43802	27456	37026	45469	34408	38799	33144	42175	42408
5	36976	44075	37928	36564	34435	28497	35569	37389	37319
4	32133	27456	34408	32516	26402	30270	30483	28288	33171
3	22460	22460	11988	21236	23431	26052	23711	25906	25839
2	18788	18788	21409	20983	21196	21715	19306	19230	21731

Tabla 4.3 Área del Bloque Multiplicador Filtros FIR Paso bajo de 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	33211	28175	26215	27945	19755	28467	25976	27027	22144
15	21994	22669	28175	26551	29562	20717	25347	27276	25816
14	29089	23065	25490	25913	17610	26375	29169	22683	22736
13	22503	23468	26595	25839	22693	15947	21728	20175	23099
12	20524	25384	19859	21738	15022	21202	22290	19915	17567
11	22214	19117	21316	20830	20687	20228	25058	19596	21059
10	24296	23731	21522	20717	19326	18881	20251	15378	20993
9	21346	18768	19094	16652	19313	18312	20308	17467	17913
8	20767	21113	19988	20717	15002	20730	17590	17051	18172
7	18269	14304	19080	17793	18941	14121	11223	17500	17380
6	19020	17370	17031	18618	11037	13771	13189	20238	19004
5	13352	12218	15794	12284	11553	10472	12208	9879	12820
4	14217	16981	17444	14064	12447	11935	7318	12484	15019
3	6107	8422	6180	10475	7777	10395	10934	9484	10222
2	10159	9820	9766	10099	11010	12524	9973	9973	13053

Tabla 4.4 Área total para implementación de Filtros FIR Paso bajo de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	112406	102979	105410	102749	78939	103271	105171	101831	101339
15	96545	85991	102726	101103	104113	84038	99898	101828	100367
14	98997	88582	95398	91429	74345	91892	99077	88199	92644
13	87767	84341	91859	91103	87957	76820	86992	85439	88362
12	81144	81613	80479	77967	60022	77432	82911	76145	78187
11	78190	70703	77292	76807	76664	71814	81034	75572	77036
10	75629	68224	72855	65211	61878	63375	71584	59872	72326
9	68035	61066	65783	63341	66002	60610	66997	64156	64602
8	62812	63158	62034	62763	45818	62776	59636	59097	60218
7	55671	47315	56482	55195	56343	47132	48625	54902	54782
6	51779	50129	49790	51376	39405	46530	45948	52996	51762
5	41467	33494	43908	40399	39667	31747	35679	37994	40935
4	37688	40452	40915	37535	29079	35406	26146	35955	38490
3	24935	27250	25008	29302	26605	29222	25118	28311	29049
2	24343	24003	23950	24283	25194	26708	24156	24156	27237

Tabla 4.5 Área del Bloque Multiplicador Filtros FIR Paso bajo de 16 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	46972	50557	40582	41663	27668	39411	40731	43193	37139
15	31491	33360	44789	38120	42784	31424	42311	38007	32103
14	36976	37621	35150	37728	28676	35273	33816	35895	35007
13	33390	31304	37142	35542	37152	31530	36946	37451	33340
12	37415	33965	39191	34078	24445	35426	40612	29046	33180
11	30183	22260	30832	29767	33643	28740	34973	28982	32449
10	33806	34667	36151	29026	23065	29561	33283	27429	28001
9	27895	24858	28683	27595	30101	25124	27885	21980	29568
8	30503	28863	28929	33789	20750	30123	23896	24073	32069
7	22649	20523	25460	25130	26518	20367	21288	19040	22526
6	24249	21465	22310	28171	19789	16871	20304	23690	26734
5	18355	14918	21212	19136	16878	10394	16146	20527	18328
4	19875	20962	19060	22709	13665	19705	16918	18132	19243
3	8748	11293	11106	12783	13322	10933	13834	16608	16099
2	12876	13029	12936	10175	13598	14107	13358	12683	13205

Tabla 4.6 Área total para implementación de Filtros FIR Paso bajo de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	126167	125362	119777	116467	86852	114215	119926	117997	116334
15	106042	96681	119341	112671	117335	94745	116863	112558	106654
14	106883	103138	105057	103244	85411	100789	103723	101411	104914
13	98654	92177	102406	100806	102416	92404	102210	102715	98604
12	98035	90195	99811	90308	69445	91655	101232	85275	93801
11	86160	73846	86809	85744	89619	80325	90950	84959	88425
10	85139	79161	87484	73520	65617	74055	84616	71923	79334
9	74584	67156	75372	74285	76790	67422	74574	68670	76257
8	72548	70908	70975	75835	51566	72169	65942	66118	74115
7	60051	53535	62862	62532	63920	53378	58691	56442	59928
6	57007	54223	55068	60929	48156	49629	53062	56449	59492
5	46469	36194	49327	47251	44993	31670	44261	48641	46443
4	43346	44434	42531	46180	30297	43176	40389	41603	42714
3	27575	30120	29934	31610	32150	29761	32661	35436	34927
2	27060	27213	27120	24359	27782	28291	27542	26867	27389

Tabla 4.7 Área del Bloque Multiplicador Filtros FIR Paso bajo de 20 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	50545	43060	55880	56582	59070	44890	57354	55478	49360
15	60803	58289	57244	58209	40233	59609	59380	50292	50731
14	51473	48931	44464	47089	37119	47261	53409	47115	51716
13	50731	42442	55072	47182	50338	43370	46926	52544	40053
12	49976	48775	48921	45894	34408	42335	41191	42099	41863
11	39175	33038	42212	45292	44368	35782	44541	36980	40921
10	45129	40220	41806	38343	32276	40023	37868	33873	44088
9	36773	32339	38300	35563	36138	32130	36534	35253	33008
8	42006	43123	36597	39534	27160	35323	31338	34581	37405
7	31162	21878	34458	29422	34518	21528	28447	27064	29226
6	29222	30726	33730	30530	26096	31355	23747	29698	32060
5	30360	15095	22763	24975	21023	13259	27057	19190	26897
4	24642	28161	32246	24512	17068	27143	24063	26691	28401
3	11915	14280	11716	16146	13775	18029	14353	13242	16991
2	13409	13731	16582	14037	17580	18472	15155	15717	16798

Tabla 4.8 Área total para implementación de Filtros FIR Paso bajo de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	125096	106382	130431	131133	133621	108211	131905	130029	123912
15	139998	133093	136439	133013	99416	134413	138575	125096	129926
14	121380	114448	114372	112605	93854	112778	123316	112632	121623
13	115995	103315	120336	112446	115602	104243	112189	117808	105317
12	110596	105004	109542	102124	79408	98565	101811	98328	102483
11	95152	84624	98189	101269	100344	87368	100517	92956	96898
10	96462	84713	93139	82837	74827	84517	89201	78367	95421
9	83463	74638	84990	82252	82827	74428	83223	81943	79697
8	84051	85169	78643	81580	57976	77369	73384	76627	79451
7	68564	54889	71860	66824	71920	54540	65849	64466	66628
6	61981	63484	66488	63288	54463	64113	56506	62456	64818
5	58475	36371	50877	53089	49138	34535	55172	47305	55012
4	48113	51632	55717	47983	33700	50615	47534	50162	51872
3	30743	33108	30543	34974	32602	36857	33181	32070	35819
2	27592	27915	30766	28221	31764	32655	29339	29901	30982

Tabla 4.9 Área del Bloque Multiplicador Filtros FIR Paso bajo de 24 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	68131	53322	71245	73354	73656	57380	67592	68012	57929
15	70397	73726	75343	72256	51020	69465	81819	71647	66631
14	64599	66382	66149	65873	53000	63471	66448	59782	62506
13	60763	56798	60920	62596	63288	53192	67722	56802	50438
12	68032	58162	59646	60091	40093	57207	55265	51569	53908
11	56120	45462	49207	56236	51240	47558	53495	48216	58365
10	58937	52630	56346	53099	39558	52254	50155	45791	49321
9	50089	36674	51889	39195	46363	41703	49813	45765	48662
8	50897	52391	49803	52042	32123	48203	47062	44491	47990
7	40299	28534	40818	38945	40200	25723	40702	29658	36148
6	29532	43253	50102	35060	25247	40436	35014	39468	38407
5	33274	18535	30786	27197	22327	13259	26571	29359	35649
4	27070	31085	33031	31295	19370	30157	30357	33098	29851
3	11915	18661	17520	16798	21881	20444	18002	17041	16489
2	17287	13731	19573	16592	21339	15518	20733	20115	20028

Tabla 4.10 Área total para implementación de Filtros FIR Paso bajo de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	142683	116644	145796	147905	148208	120702	142144	142563	132481
15	149592	148530	154538	147060	110204	144269	161014	146451	145826
14	134506	131898	136056	131389	109735	128988	136356	125299	132414
13	126027	117671	126184	127860	128552	114066	132986	122066	115702
12	128652	114392	120266	116321	85093	113437	115885	107799	114528
11	112096	97048	105184	112213	107217	99143	109472	104193	114342
10	110270	97124	107679	97593	82109	96748	101488	90285	100654
9	96778	78972	98578	85884	93053	84002	96502	92454	95351
8	92943	94436	91849	94087	62939	90249	89108	86536	90036
7	77701	61545	78220	76348	77602	58734	78104	67060	73550
6	62290	76012	82861	67819	53615	73194	67772	72226	71165
5	61389	39810	58901	55311	50442	34535	54686	57474	63764
4	50541	54556	56502	54766	36002	53628	53828	56569	53322
3	30743	37489	36348	35626	40708	34628	36830	35869	35316
2	31471	27915	33756	30776	35523	34345	34917	34299	34212

Tabla 4.11 Coeficientes nulos en Bloque Multiplicador FIR Paso bajo precisión 8 bit

orden\frec	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
2									
3							2		
4					2				
5				2				2	
6			2		2		2		
7			2	2		2		2	2
8					4		2	2	
9		2	2	4			2	2	2
10		2		2	4	2		2	
11		4		4	2	2	2	2	4
12		4	2	2	6	2	2	4	
13		4	2	4	2	2	4	4	6
14		4	4	4	8	4	4	6	2
15	2	2	4	4	2	4		6	6
16	2	4	4	6	8	6	2	4	2

Tabla 4.12 Coeficientes nulos Bloque Multiplicador FIR Paso bajo precisión mayor 12 bit

orden\frec	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
2									
3									
4					2				
5				2				2	
6					2				
7				2				2	
8					4				
9				2				2	
10		2		2	4	2		2	
11				2				2	
12		2		2	6	2		2	
13				2				2	
14		2		2	6	2		2	
15				4				4	
16		2		2	8	2		2	

Tabla 4.13 Grafos que requieren replicación en Filtros FIR Paso bajo de 8 bit

frec\orden	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
0,9			R2												
0,8		R2						R3		R3		R2			
0,7		R2			R1							R1			
0,6															
0,5													R2	R1	
0,4	R2														R2
0,3	R3												R2		
0,2										R2	R1				
0,1															R2

Tabla 4.14 Grafos que requieren replicación en Filtros FIR Paso bajo de 12 bit

frec\orden	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
0,9			R4	R2								R4	R2		
0,8	R1							R1	R2	R1					
0,7		R2	R3				R2				R2	R3			
0,6	R2									R2					
0,5															
0,4	R1			R1					R1	R1			R1		
0,3		R2				R2			R2		R2				R2
0,2			R1	R5								R1	R5		
0,1					R2				R2					R2	

Tabla 4.15 Grafos que requieren replicación en Filtros FIR Paso bajo de 16 bit

frec\orden	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
0,9	R2						R4		R2			R9		R4	R2
0,8	R1								R2			R8		R6	R6
0,7	R2													R6	
0,6	R2	R2							R3	R4					R4
0,5										R9			R6	R7	R2
0,4		R1										R4		R5	R7
0,3	R2												R2	R2	R6
0,2				R2		R2							R9	R6	R6
0,1				R2										R8	R6

Tabla 4.16 Grafos que requieren replicación en Filtros FIR Paso bajo de 20 bit

frec\orden	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
0,9				R3									R3		
0,8					R1	R1		R4						R1	R1
0,7						R1	R2		R2						R1
0,6	R4			R4			R3			R4			R4		
0,5				R4		R1		R2					R4		R1
0,4	R1		R2	R2						R1		R2	R2		
0,3				R3									R3		
0,2				R6									R6		
0,1					R8		R5							R8	

Tabla 4.17 Grafos que requieren replicación en Filtros FIR Paso bajo de 24 bit

frec\orden	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
0,9								R1							
0,8			R4									R4			
0,7					R1			R2						R1	
0,6															
0,5		R4				R2					R4				R2
0,4		R2					R4				R2				
0,3		R1									R1				
0,2	R2						R4			R2					
0,1	R2	R1			R14	R1			R1	R2	R1			R14	R1

Tabla 4.18 Área Registro Check Filtros FIR DetectError Paso bajo de 8 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	31834	27443	31834	23052	18661	23052	27443	27443	31834
15	20857	20857	34029	25247	29638	25247	25247	29638	29638
14	27443	18661	23052	23052	14270	23052	23052	23052	31834
13	16466	20857	20857	25247	25247	20857	25247	20857	29638
12	27443	18661	23052	23052	14270	23052	23052	18661	27443
11	16466	20857	20857	20857	20857	16466	25247	16466	25247
10	23052	18661	23052	18661	14270	18661	23052	18661	23052
9	16466	16466	16466	20857	20857	12075	16466	16466	20857
8	18661	14270	14270	18661	9879	18661	18661	18661	18661
7	12075	12075	16466	12075	16466	12075	12075	16466	16466
6	14270	14270	9879	14270	9879	14270	9879	14270	14270
5	12075	7684	12075	12075	12075	7684	12075	12075	12075
4	9879	9879	9879	9879	5489	9879	9879	9879	9879
3	7684	7684	3293	7684	7684	7684	7684	7684	7684
2	5489	5489	5489	5489	5489	5489	5489	5489	5489

Tabla 4.19 Área Registro Check Filtros FIR DetectError Paso bajo mayor de 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	36224	31834	36224	31834	18661	31834	36224	31834	36224
15	34029	25247	34029	34029	34029	25247	34029	34029	34029
14	31834	27443	31834	27443	18661	27443	31834	27443	31834
13	29638	25247	29638	29638	29638	25247	29638	29638	29638
12	27443	23052	27443	23052	14270	23052	27443	23052	27443
11	25247	20857	25247	25247	25247	20857	25247	25247	25247
10	23052	18661	23052	18661	14270	18661	23052	18661	23052
9	20857	16466	20857	20857	20857	16466	20857	20857	20857
8	18661	18661	18661	18661	9879	18661	18661	18661	18661
7	16466	12075	16466	16466	16466	12075	16466	16466	16466
6	14270	14270	14270	14270	9879	14270	14270	14270	14270
5	12075	7684	12075	12075	12075	7684	12075	12075	12075
4	9879	9879	9879	9879	5489	9879	9879	9879	9879
3	7684	7684	7684	7684	7684	7684	7684	7684	7684
2	5489	5489	5489	5489	5489	5489	5489	5489	5489

Tabla 4.20 Área del Bloque Multiplicador DetectError Filtros FIR Paso bajo de 8 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	14316	16502	17170	14097	10624	14858	20596	13278	9417
15	12560	15188	20679	12357	16435	10561	11389	8445	11678
14	18627	10019	12357	14822	8698	15387	12450	12906	11845
13	10275	12710	11436	12267	12547	10228	10115	9207	14020
12	18887	15048	15348	13841	11043	12217	12434	9802	14483
11	12803	12876	9287	10877	9999	13611	12520	10401	11396
10	12740	12574	12893	12288	11227	12358	12640	9630	11612
9	10182	12783	12667	8848	13033	9118	10874	9650	9204
8	13395	11080	11150	9953	8818	12817	13049	6799	9417
7	11366	10990	8163	6041	12740	8679	9647	12058	9650
6	11044	6180	11107	12710	6041	6041	7225	11825	9650
5	8862	13725	9813	8449	6320	7222	7454	9274	9204
4	8662	6180	10937	9044	12178	6799	9207	4817	9700
3	3632	3632	0	2408	4604	7225	4883	7079	7012
2	4604	4604	7225	6799	7012	9939	5123	5046	9743

Tabla 4.21 Área total requerida por Filtros FIR DetectError Paso bajo de 8 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	118506	111910	114521	98275	88468	99036	113556	108686	113606
15	87451	92527	122421	98477	113785	99129	97509	105795	109029
14	109138	78070	91639	96552	70416	97117	91731	94636	113586
13	71487	87601	83878	95939	96219	85119	93787	81650	108922
12	106950	80651	92181	93122	70313	91499	89267	75406	102546
11	71567	85319	79281	80871	79993	74824	93744	69166	92619
10	87125	75729	87278	75443	68048	75513	87025	72785	85997
9	66498	71548	68983	76394	80579	56652	69638	65966	76750
8	74102	60557	60627	70659	49513	73523	73756	67506	70124
7	54004	56076	62031	48679	66608	53765	52284	65926	63518
6	58072	41726	46906	59739	44288	53069	43024	58854	56679
5	49051	54167	50002	48639	46510	36181	47644	49464	49394
4	42012	37336	44288	42395	34299	40150	42558	38167	43050
3	30144	30144	15281	28920	31115	33736	31395	33590	33523
2	24276	24276	26897	26471	26684	29612	24795	24718	29415

Tabla 4.22 Área del Bloque Multiplicador DetectError Filtros FIR Paso bajo de 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	33211	28175	28830	30420	19755	28467	25976	27027	22144
15	24190	22669	28175	26551	29562	20717	25347	29472	28225
14	29089	25261	28511	25913	17610	26375	31714	22683	22736
13	25141	23468	26595	25839	22693	15947	21728	20175	23099
12	20524	25384	19859	21738	15022	21202	22290	19915	17567
11	22214	19117	21316	23468	20687	20228	25058	19596	23255
10	24296	26674	21522	20717	19326	21901	20251	15378	24243
9	21346	18768	19094	21256	19313	18312	20308	17467	17913
8	20767	21113	19988	20717	17410	20730	17590	17051	20963
7	25723	21379	19080	20507	18941	16529	11223	17500	17380
6	19020	17370	17031	18618	13445	13771	13189	20238	19004
5	13352	14626	15794	12284	16156	15285	12208	9879	12820
4	16762	16981	17444	16259	19313	11935	9513	12484	15019
3	8303	13026	8589	15291	9973	15208	16210	9484	10222
2	10159	9820	9766	14769	16363	19390	14364	14363	20524

Tabla 4.23 Área total requerida por Filtros FIR DetectError Paso bajo de 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	148630	134812	144249	137058	97600	135105	141395	133665	137563
15	132770	111238	136755	135132	138142	109286	133928	138052	136805
14	130831	118220	130252	118872	93006	119335	133455	115642	124477
13	120043	109588	121497	120742	117595	102067	116630	115077	118001
12	108587	104665	107922	101019	74292	100484	110353	99197	105630
11	103438	91559	102540	104692	101911	92670	106282	100820	104479
10	98681	89829	95907	83872	76148	85056	94636	78533	98628
9	88891	77532	86639	88802	86859	77076	87854	85013	85459
8	81474	81819	80695	81424	58106	81437	78297	77758	81670
7	79591	66465	72948	74375	72808	61615	65091	71368	71248
6	66049	64399	64060	65647	51692	60800	60218	67266	66032
5	53542	43586	55983	52474	56346	44244	47754	50069	53010
4	50112	50332	50794	49610	41434	45286	38220	45834	48369
3	34814	39538	35100	41803	36484	41720	38077	35995	36733
2	29831	29492	29439	34442	36035	39062	34036	34036	40196

Tabla 4.24 Área del Bloque Multiplicador DetectError Filtros FIR Paso bajo de 16 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	50006	54036	43219	44071	27668	39411	43368	43193	37139
15	31491	33360	44789	40315	42784	33898	42311	38007	32103
14	36976	37621	35150	37728	28676	35273	33816	35895	35007
13	33390	31304	37142	35542	37152	31530	36946	39855	36589
12	37415	33965	39191	34078	24445	35426	40612	29046	33180
11	30183	22260	30832	29767	33643	28740	34973	31945	32449
10	39098	34667	36151	29026	23065	29561	33283	27429	28001
9	27895	24858	28683	27595	30101	25124	27885	21980	29568
8	32698	31547	28929	36426	20750	30123	23896	24073	32069
7	22649	20523	25460	30498	40289	20367	21288	19040	22526
6	24249	21465	22310	28171	19789	16871	20304	23690	26734
5	28853	25169	21212	19136	16878	15207	16146	20527	18328
4	19875	20962	19060	22709	21365	19705	19532	18132	19243
3	13351	18441	18114	12783	20680	16055	16548	27252	26234
2	15071	20247	12936	14845	16236	22326	20829	19478	20676

Tabla 4.25 Área total requerida por Filtros FIR DetectError Paso bajo de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	165425	160675	158638	150709	105513	146049	158787	149831	152558
15	140071	121928	153370	148895	151364	122466	150892	146587	140683
14	138717	130581	136891	130687	104072	128232	135557	128854	136748
13	128292	117424	132044	130444	132054	117651	131848	134757	131491
12	125478	113247	127254	113360	83715	114707	128675	108327	121244
11	111407	94703	112056	110991	114866	101182	116197	113169	113672
10	113483	97822	110536	92181	79887	92716	107668	90584	102386
9	95441	83622	96229	95142	97646	83888	95431	89527	97114
8	93404	92253	89636	97133	61445	90830	84603	84779	92776
7	76517	65610	79328	84366	94157	65453	75157	72908	76394
6	71277	68493	69338	75199	58036	63899	67332	70719	73762
5	69042	54129	61402	59326	57068	44167	56336	60716	58518
4	53225	54313	52410	56059	43486	53055	52882	51482	52593
3	39862	44952	44626	39294	47192	42567	43059	53764	52746
2	34744	39920	32609	34518	35908	41999	40502	39151	40349

Tabla 4.26 Área del Bloque Multiplicador DetectError Filtros FIR Paso bajo de 20 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	50545	43060	55880	60214	59070	44890	57354	55478	49360
15	60803	58289	57244	58209	40233	59609	59380	55824	50731
14	51473	48931	44464	47089	37119	50155	56093	47115	54889
13	57510	42442	55072	53592	50338	43370	52434	52544	40053
12	49976	48775	48921	48090	34408	42335	41191	45202	41863
11	39175	33038	45921	48389	44368	35782	44541	36980	40921
10	45129	40220	41806	41899	32276	40023	37868	33873	44088
9	36773	32339	38300	44231	36138	32130	36534	35253	33008
8	42006	43123	36597	39534	37595	35323	37894	34581	37405
7	31162	35925	34458	29422	37552	24472	28447	27064	29226
6	29222	30726	33730	30530	26096	31355	23747	29698	32060
5	44391	25557	35652	24975	24120	20477	27057	19190	43739
4	24642	28161	52471	37475	28018	44634	38147	26691	28401
3	14783	24183	19147	26099	21509	30267	22437	20597	27636
2	20624	21346	27123	16512	28966	30673	19686	22513	27479

Tabla 4.27 Área total requerida por Filtros FIR DetectError Paso bajo de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	161320	138215	166656	166599	152283	140045	168130	161863	160136
15	174027	158340	170468	167042	133445	159661	172604	164657	163955
14	153214	141891	146205	140048	112515	143115	157834	140075	156630
13	152412	128562	149974	148494	145241	129490	147336	147446	134955
12	138039	128056	136984	127371	93678	121617	129254	124484	129926
11	120399	105480	127145	129613	125592	108224	125765	118204	122145
10	119514	103375	116191	105054	89098	103178	112253	97028	118473
9	104319	91103	105846	111777	103684	90894	104080	102799	100554
8	102713	103830	97304	100241	78290	96030	98601	95288	98112
7	85029	81011	88326	83290	91419	69558	82315	80931	83093
6	76251	77755	80758	77558	64343	78383	70776	76727	79088
5	84580	54516	75842	65164	64309	49437	67247	59380	83928
4	57992	61512	85821	70826	50139	77984	71498	60042	61751
3	41294	50694	45658	52610	48020	56778	48948	47108	54147
2	40296	41018	46796	36185	48639	50345	39358	42185	47152

Tabla 4.28 Área del Bloque Multiplicador DetectError Filtros FIR Paso bajo de 24 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	68131	53322	71245	73354	73656	57380	67592	68012	57929
15	70397	73726	82624	72256	51020	69465	81819	71647	66631
14	64599	66382	66149	65873	53000	63471	66448	62879	62506
13	60763	56798	60920	62596	63288	53192	67722	56802	50438
12	68032	64822	59646	60091	40093	60032	55265	51569	53908
11	56120	47870	49207	56236	51240	47558	58964	48216	58365
10	58937	52630	56346	53099	39558	52254	50155	45791	49321
9	53109	36674	51889	39195	46363	41703	56861	45765	48662
8	54377	56253	49803	52042	51270	48203	47062	44491	52158
7	40299	48442	40818	43825	40200	35090	40702	29658	36148
6	29532	43253	50102	35060	41963	40436	35014	39468	38407
5	33274	32249	50791	45831	36554	20477	43546	47973	35649
4	45625	53469	53123	53645	26901	49723	38955	33098	49264
3	14783	32712	30523	27326	37186	34235	23890	27429	26691
2	28228	21346	32722	26837	36254	25071	29658	33653	27882

Tabla 4.29 Área total requerida por Filtros FIR DetectError Paso bajo de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	178907	148477	182021	179739	166869	152535	178368	174397	168705
15	183621	173778	195848	181089	144233	169517	195043	180480	179855
14	166340	159341	167890	158832	128396	156431	168189	155839	164248
13	155666	142919	155822	157498	158190	139313	162624	151704	145340
12	156095	144103	147709	139373	99363	139313	143328	130851	141971
11	137344	120313	130431	137460	132464	120000	140188	129440	139589
10	133322	115785	130731	116254	96379	115409	124540	108946	123705
9	120655	95438	119434	106741	113909	100467	124407	113310	116208
8	115083	116960	110510	112748	91965	108910	107769	105197	112865
7	94167	93528	94686	97693	94067	80176	94570	83526	90016
6	76560	90282	97131	82089	80209	87464	82042	86496	85435
5	73464	61209	90980	86021	76743	49437	83735	88163	75839
4	78975	86819	86473	86995	49021	83074	72306	66448	82614
3	41294	59223	57034	53838	63697	56103	50402	53941	53202
2	47900	41018	52394	46510	55927	49387	49331	53326	47554

Tabla 4.30 Área del Comparador Filtros FIR DMR Paso bajo de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	16466	12075	18661	12075	12075	16466	16466	16466	18661
15	14270	9879	14270	14270	9879	14270	16466	14270	16466
14	12075	9879	12075	12075	7684	12075	14270	12075	14270
13	12075	9879	12075	9879	7684	9879	12075	9879	12075
12	9879	7684	7684	9879	5489	9879	9879	9879	9879
11	7684	7684	5489	7684	5489	7684	7684	7684	7684
10	5489	5489	5489	5489	3293	5489	5489	5489	5489
9	3293	3293	3293	3293	3293	3293	3293	3293	3293
8	16466	12075	18661	12075	12075	16466	16466	16466	18661
7	14270	9879	14270	14270	9879	14270	16466	14270	16466
6	12075	9879	12075	12075	7684	12075	14270	12075	14270
5	12075	9879	12075	9879	7684	9879	12075	9879	12075
4	9879	7684	7684	9879	5489	9879	9879	9879	9879
3	7684	7684	5489	7684	5489	7684	7684	7684	7684
2	5489	5489	5489	5489	3293	5489	5489	5489	5489

Tabla 4.31 Área del Comparador Filtros FIR DMR Paso bajo mayor de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	18661	16466	18661	16466	12075	16466	18661	16466	18661
15	16466	14270	16466	14270	9879	14270	16466	14270	16466
14	14270	12075	14270	12075	7684	12075	14270	12075	14270
13	12075	9879	12075	9879	7684	9879	12075	9879	12075
12	9879	9879	9879	9879	5489	9879	9879	9879	9879
11	7684	7684	7684	7684	5489	7684	7684	7684	7684
10	5489	5489	5489	5489	3293	5489	5489	5489	5489
9	3293	3293	3293	3293	3293	3293	3293	3293	3293
8	18661	16466	18661	16466	12075	16466	18661	16466	18661
7	16466	14270	16466	14270	9879	14270	16466	14270	16466
6	14270	12075	14270	12075	7684	12075	14270	12075	14270
5	12075	9879	12075	9879	7684	9879	12075	9879	12075
4	9879	9879	9879	9879	5489	9879	9879	9879	9879
3	7684	7684	7684	7684	5489	7684	7684	7684	7684
2	5489	5489	5489	5489	3293	5489	5489	5489	5489

Tabla 4.32 Área del Bloque Multiplicador de Filtros FIR DMR Paso bajo de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	28632	33004	34340	28194	21248	24900	31986	26556	18834
15	25120	25560	36542	24714	32870	21122	22778	16890	23356
14	32864	20038	24714	29644	17396	30774	20510	25812	23690
13	20550	25420	22872	24534	25094	20456	20230	18414	28040
12	37774	30096	30696	27682	22086	24434	24868	19604	28966
11	25606	25752	18574	21754	19998	27222	25040	20802	22792
10	25480	25148	25786	24575	22453	24715	25281	19260	23225
9	20364	20750	25334	17696	26066	18235	21748	19300	18408
8	26791	22160	22300	19905	17637	25633	26099	13598	18834
7	22733	17590	16326	12081	25480	17357	19293	19300	19300
6	22087	12361	22214	25420	12081	12081	14450	18834	19300
5	17723	22633	19626	16898	12640	14443	14909	18548	18408
4	17324	12361	21874	18089	19539	13598	14024	9633	19400
3	7265	7265	0	4817	9207	14450	9766	14157	14024
2	9207	9207	14450	13598	14024	15062	10245	10092	15095

Tabla 4.33 Área Total de Filtros FIR DMR Paso bajo de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	117454	115239	116323	101395	90310	98101	111773	108791	107655
15	90131	93019	121818	98760	115950	97616	96824	99970	106437
14	110202	79307	93019	100397	72528	101527	88814	96565	110063
13	74078	90431	85434	96132	96692	85467	91828	80977	108672
12	112664	86917	96552	95986	74770	92739	90724	76426	103856
11	76686	88315	78688	81868	80112	80751	94190	71883	91941
10	88888	79521	89194	78949	72688	79088	88688	73633	86633
9	68996	71830	73966	75363	83732	60281	72828	67932	76075
8	78716	65051	65191	71830	53941	77558	78024	65523	70759
7	59882	57187	62510	49231	71664	56955	56442	65484	65484
6	62530	41321	53622	65863	45938	52524	45858	59276	59742
5	52424	59782	54327	51599	47341	40110	49610	53249	53109
4	46284	39125	50834	47049	39464	42558	42984	38593	48359
3	30483	30483	14184	28035	32426	37668	32985	37375	37242
2	26684	26684	31927	31075	31501	32539	27722	27569	32572

Tabla 4.34 Área del Bloque Multiplicador de Filtros FIR DMR Paso bajo de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	66422	56349	52431	55890	39511	56935	51952	54054	44288
15	43988	45339	56349	53103	59123	41434	50694	54553	51632
14	58179	46131	50980	51825	35220	52750	58338	45365	45472
13	45006	46936	53189	51679	45385	31894	43456	40349	46197
12	41048	50768	39717	43476	30044	42405	44580	39830	35133
11	44427	38234	42631	41660	41374	40456	50116	39192	42119
10	48592	47461	43044	41434	38653	37761	40502	30756	41986
9	42691	37535	38187	33304	38626	36624	40615	34934	35825
8	41533	42225	39977	41434	30004	41460	35180	34102	36344
7	36537	28607	38160	35586	37881	28241	22447	35000	34761
6	38041	34741	34062	37236	22074	27543	26378	40476	38007
5	26704	24436	31587	24569	23105	20943	24416	19759	25640
4	28434	33963	34887	28128	24895	23870	14636	24968	30037
3	12215	16845	12361	20950	15554	20790	21868	18967	20444
2	20318	19639	19533	20198	22021	25048	19945	19945	26106

Tabla 4.35 Área Total de Filtros FIR DMR Paso bajo de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	164278	147619	150287	147160	108574	148204	149808	145324	142144
15	136103	121833	148464	145217	151238	117928	142809	146668	143747
14	144552	125918	137354	131612	101834	132537	144712	125152	131845
13	125638	120981	133821	132311	126017	105939	124088	120981	126829
12	115938	119072	114608	111780	82728	110709	119471	108135	110024
11	113577	100797	111780	110809	110523	103019	119265	108341	111268
10	112000	101834	106451	95807	88888	92135	103910	85129	105394
9	100357	88615	95854	90970	96293	87704	98282	92600	93492
8	93459	94150	91902	93359	66308	93385	87105	86027	88269
7	82721	68205	84344	81770	84065	67839	68630	81184	80945
6	78483	75183	74505	77678	55930	67985	66821	80918	78450
5	61405	50102	66289	59270	57806	46610	54473	54460	60341
4	57394	62922	63847	57088	44820	52830	38952	53928	58997
3	35433	40063	35579	44168	38773	44008	40442	42185	43662
2	37795	37116	37010	37675	39498	42525	37422	37422	43582

Tabla 4.36 Área del Bloque Multiplicador de Filtros FIR DMR Paso bajo de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	93944	101114	81164	83326	55336	78822	81462	86386	74278
15	62982	66720	89578	76240	85568	62848	84622	76014	64206
14	73952	75242	70300	75456	57352	70546	67632	71790	70014
13	66780	62608	74284	71084	74304	63060	73892	74902	66680
12	74830	67930	78382	68156	48890	70852	81224	58092	66360
11	60366	44520	61664	59534	67286	57480	69946	57964	64898
10	67612	69334	72302	58052	46131	59122	66566	54858	56002
9	55790	49716	57366	55190	60201	50248	55770	43960	59136
8	61006	57726	57858	67578	41500	60246	47792	48146	64138
7	45298	41046	50920	50260	53036	40734	42576	38080	45052
6	48498	42930	44620	56342	39578	33742	40608	47380	53468
5	36710	29836	42424	38272	33756	20788	32292	41054	36656
4	39750	41924	38120	45418	27330	39410	33836	36264	38486
3	17496	22586	22212	25566	26644	21866	27668	33216	32198
2	25752	26058	25872	20350	27197	28214	26716	25366	26410

Tabla 4.37 Área Total de Filtros FIR DMR Paso bajo de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	191800	192385	179020	174596	124399	170092	179318	177656	172134
15	155096	143214	181693	168354	177682	139342	176737	168128	156320
14	160325	155029	156673	155242	123966	150332	154005	151576	156387
13	147412	136654	154916	151716	154936	137107	154524	155534	147312
12	149720	136235	153272	136461	101574	139156	156114	126396	141251
11	129516	107083	130814	128684	136435	120042	139096	127114	134047
10	131020	123707	135710	112425	96366	113495	129974	109231	119410
9	113456	100796	115032	112857	117868	101328	113436	101627	116802
8	112930	109650	109783	119503	77804	112171	99717	100070	116063
7	91482	80644	97104	96444	99220	80331	88761	84264	91236
6	88940	83372	85062	96784	73434	74184	81050	87823	93910
5	71410	55503	77125	72973	68457	46455	66993	75754	71357
4	68710	70885	67080	74378	47255	68370	62796	65224	67446
3	40714	45804	45431	48784	49863	45085	50886	56435	55417
2	43229	43535	43349	37827	44674	45691	44193	42843	43887

Tabla 4.38 Área del Bloque Multiplicador de Filtros FIR DMR Paso bajo de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	101089	86121	111760	113164	118140	89780	114708	110955	98721
15	121607	116577	114488	116417	80466	119218	118759	100584	101462
14	102945	97863	88928	94177	74239	94523	106817	94230	103431
13	101462	84883	110144	94363	100677	86739	93851	105088	80106
12	99952	97550	97843	91789	68817	84670	82382	84198	83725
11	78350	66076	84424	90585	88735	71564	89081	73959	81843
10	90259	80439	83612	76687	64552	80047	75735	67745	88176
9	73547	64679	76600	71125	72276	64259	73068	70506	66016
8	84012	86247	73194	79069	54320	70646	62676	69163	74811
7	62323	43755	68916	58844	69036	43057	56895	54127	58452
6	58445	61452	67459	61059	52191	62709	47494	59396	64120
5	60720	30190	45525	49949	42046	26518	54114	38380	53795
4	49284	56323	64492	49024	34136	54287	48126	53382	56802
3	23830	28560	23431	32293	27549	36058	28707	26485	33983
2	26817	27463	33164	28075	35160	36943	30310	31434	33597

Tabla 4.39 Área Total de Filtros FIR DMR Paso bajo de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	194302	165908	204973	204181	202571	169567	207920	201972	191933
15	218365	204554	211246	208785	157212	207195	215517	192951	198220
14	189319	177650	175301	173964	140853	174310	193191	174017	189804
13	182094	158929	190776	174995	181309	160785	174483	185720	160738
12	174842	165854	172733	160093	121500	152974	157272	152502	158616
11	147499	128639	153573	159734	157884	134127	158230	143108	150992
10	153666	134812	147020	131060	114787	134420	139143	122119	151584
9	131213	115759	134267	128792	129942	115340	130734	128173	123682
8	135937	138172	125119	130994	90624	122571	114601	121088	126736
7	108507	83353	115100	105028	115220	82654	103078	100311	104635
6	98887	101894	107902	101502	86047	103152	87937	99839	104562
5	95421	55857	80226	84650	76747	52185	88815	73081	88496
4	78244	85282	93452	77984	54061	83246	77086	82342	85761
3	47049	51779	46649	55511	50768	59276	51925	49703	57201
2	44294	44940	50641	45552	52637	54420	47787	48911	51074

Tabla 4.40 Área del Bloque Multiplicador de Filtros FIR DMR Paso bajo de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	136263	106644	142490	146708	147313	114761	135185	136023	115859
15	140793	147453	150686	144512	102041	138930	163639	143295	133262
14	129197	132763	132298	131745	105999	126942	132896	119564	125013
13	121527	113597	121839	125192	126576	106385	135444	113603	100876
12	136063	116324	119291	120183	80186	114415	110530	103138	107815
11	112239	90924	98415	112472	102480	95115	106990	96432	116730
10	117874	105261	112692	106199	79115	104509	100311	91582	98641
9	100178	73347	103777	78390	92727	83406	99626	91529	97324
8	101794	104782	99606	104083	64246	96406	94124	88981	95980
7	80599	57068	81637	77891	80399	51446	81404	59316	72296
6	59064	86506	100204	70121	50495	80871	70027	78935	76813
5	66548	37069	61572	54393	44654	26518	53143	58718	71298
4	54140	62170	66062	62590	38739	60314	60713	66195	59702
3	23830	37322	35040	33597	43762	40888	36005	34082	32978
2	34575	27463	39145	33184	42678	31035	41467	40229	40057

Tabla 4.41 Área Total de Filtros FIR DMR Paso bajo de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	229475	186431	235702	237725	231744	194548	228397	227040	209071
15	237552	235429	247444	236880	178787	226907	260397	235662	230021
14	215571	212550	218671	211532	172614	206729	219270	199351	211386
13	202159	187642	202471	205824	207208	180431	216076	194235	181508
12	210954	184629	194182	188487	132870	182719	185420	171443	182706
11	181389	153487	167564	181621	171629	157678	176140	165582	185879
10	181282	159634	176100	160572	129350	158882	163719	145956	162049
9	157844	124427	161443	136056	150393	134486	157292	149196	154990
8	153720	156707	151531	156008	100550	148331	146049	140906	147905
7	126782	96665	127820	124075	126583	91044	127587	105500	118480
6	99506	126949	140647	110563	84351	121314	110470	119378	117256
5	101249	62736	96273	89094	79355	52185	87844	93419	105999
4	83100	91130	95022	91549	58664	89274	89673	95155	88662
3	47049	60540	58259	56815	66980	59463	59223	57301	56196
2	52052	44940	56622	50661	60155	53156	58944	57706	57533

Tabla 4.42 Área del Bloque Multiplicador Filtros FIR Paso alto de 8 bits

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	23644	23464	18595	24675	15351	14264	23564	18857	25124,30
14	23331	22543	21412	21628	12308	22859	21139	19024	21182,52
12	22014	22114	18651	10022	14985	17227	17068	15913	17091,04
10	19406	18335	19772	17896	15568	16499	14955	15847	17836,16
8	13103	15661	13399	11227	11247	12684	12990	14733	11645,73
6	10212	15132	15721	9706	13006	15734	14520	15142	13242,40
4	13788	3819	6393	11037	8655	7405	12617	10139	9952,59
2	11110	8911	5968	6107	6227	8682	3632	3632	3632,43

Tabla 4.43 Área total para implementación de Filtros FIR Paso alto de 8 bits

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	134673	121321	131818	122531	97587	109419	134593	111817	133704
14	123227	109359	118763	114588	78416	118304	118910	95857	116085
12	105686	97004	102323	89304	74255	96509	98292	85908	98315
10	95987	84105	94157	81051	72389	79654	91979	79002	92221
8	64522	67216	65024	62646	53883	64103	57570	66152	63065
6	66528	71447	67645	66022	53701	72050	63996	64618	69558
4	47138	39364	39743	44387	30776	40755	45967	43489	43303
2	30783	28584	27835	25780	28308	28355	23305	23305	23305

Tabla 4.44 Área del Bloque Multiplicador Filtros FIR Paso alto de 12 bits

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	36368	33261	36101	36181	20823	33084	37997	29319	40629
14	39341	34365	31900	29375	21346	23601	35772	30972	36224
12	31870	27586	28501	26491	23960	30753	30107	27200	30563
10	28660	27746	27805	27137	18042	30280	25849	27785	31262
8	24722	24429	27742	23368	18229	21485	22666	25015	27240
6	19210	23687	24007	18791	16379	22792	23375	20840	20191
4	18155	14822	13748	14164	14034	16067	18352	16313	16230
2	9144	9051	11390	11010	8728	14470	3632	11759	6014

Tabla 4.45 Área total para implementación de Filtros FIR Paso alto de 12 bits

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	115562	108065	115296	110985	80007	107888	117192	104123	119824
14	109249	99882	101808	94892	78081	89118	105680	96489	106132
12	92491	83815	89121	82721	68960	86982	90728	83429	91183
10	79993	72239	79138	71631	60594	74774	77182	72279	82595
8	66768	66475	69788	65414	49044	63531	64712	67060	69286
6	51968	56446	56765	51549	44747	55551	56133	53598	52950
4	41627	38294	37219	37635	30666	39538	41823	39784	39701
2	23328	23235	25573	25194	22912	28654	17816	25943	20198

Tabla 4.46 Área del Bloque Multiplicador Filtros FIR Paso alto de 16 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	50165	46074	53202	47721	31900	44358	54859	39681	57041
14	51938	45821	47727	47840	31464	43982	42042	32599	52058
12	46347	45392	44647	37898	32293	32911	35293	36234	40828
10	38057	35832	33454	38353	27047	35839	35010	36660	44125
8	31910	33417	32891	30776	27955	30796	31814	34545	35885
6	24908	31108	28793	24133	19469	26538	28062	26950	28727
4	20876	19203	20537	17906	17597	21691	24968	21705	22347
2	9237	12108	11390	11709	11662	15478	6177	11988	14071

Tabla 4.47 Área total para implementación de Filtros FIR Paso alto de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	129360	120878	132397	122525	91083	119162	134054	114485	136236
14	121846	111338	117635	113357	88199	109498	111950	98115	121966
12	106967	101622	105267	94127	77292	89141	95913	92464	101449
10	89390	80326	84787	82847	69598	80333	86343	81154	95458
8	73956	75463	74937	72822	58771	72842	73859	76590	77931
6	57666	63867	61552	56891	47837	59296	60820	59709	61485
4	44348	42674	44008	41377	34229	45163	48439	45176	45818
2	23421	26292	25573	25893	25846	29662	20361	26172	28254

Tabla 4.48 Área del Bloque Multiplicador Filtros FIR Paso alto de 20 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	69598	63621	66491	59805	41154	56283	66847	56921	70776
14	68960	49450	59865	56163	37206	50445	58262	48017	56702
12	58887	53778	53472	49666	37878	49211	48392	47022	56436
10	46127	39494	50205	44301	37226	45289	47448	42787	56020
8	41813	43227	40323	41989	29485	37548	41500	45412	45978
6	37944	38423	39275	30061	25580	30896	36703	36105	37163
4	26924	32083	24722	28022	21236	24166	24682	28657	25540
2	12108	12108	21282	17450	12873	15258	12015	18275	18106

Tabla 4.49 Área total para implementación de Filtros FIR Paso alto de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	148793	138425	145686	134609	100338	131087	146042	131725	149971
14	138867	114967	129773	121680	93941	115962	128170	113533	126609
12	119508	110007	114092	105896	82877	105440	109013	103251	117056
10	97460	83988	101538	88795	79777	89783	98781	87281	107353
8	83859	85272	82368	84035	60301	79594	83546	87458	88023
6	70703	71182	72033	62819	53948	63654	69462	68863	69921
4	50395	55554	48193	51493	37868	47637	48153	52128	49011
2	26292	26292	35466	31634	27057	29442	26199	32459	32289

Tabla 4.50 Área del Bloque Multiplicador Filtros FIR Paso alto de 24 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	82046	75183	92980	68760	48579	70769	83602	67998	87750
14	82644	69153	71697	69326	48589	67559	76693	61172	76787
12	66265	67400	62796	61033	45096	57454	66182	57959	68224
10	62769	56216	63168	55255	45299	55411	63108	53279	73054
8	51253	54077	53948	49457	40243	52271	49031	60012	51659
6	45625	46706	40007	41058	27709	46636	45562	41909	43466
4	39684	27619	28504	36098	22031	35459	31814	38047	34768
2	20085	12108	14350	18032	16326	18957	18358	14696	18834

Tabla 4.51 Área total para implementación de Filtros FIR Paso alto de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	161241	149987	172174	143564	107762	145573	162797	142802	166945
14	152552	134669	141605	134842	105324	133076	146601	126689	146694
12	126886	123629	123416	117262	90096	113683	126802	114189	128845
10	114102	100710	114501	99749	87850	99905	114441	97773	124387
8	93299	96123	95993	91503	71059	94317	91077	102057	93705
6	78383	79464	72765	73816	56076	79395	78320	74668	76224
4	63155	51090	51975	59569	38663	58931	55285	61518	58239
2	34269	26292	28534	32216	30510	33141	32542	28880	33018

Tabla 4.52 Coeficientes nulos en Bloque Multiplicador Paso alto precisión 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	2	6		6	6	2	2	2	
14	2	6	2	2	6	2		2	
12	2	4	2	2	6	2		2	
10		2		2	4	2		2	
8		2	2		4				
6			2		2				
4					2				
2									

Tabla 4.53 Coeficientes nulos en Bloque Multiplicador Paso alto precisión mayor 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16		2		2	6	2		2	
14		2		2	6	2		2	
12		2		2	6	2		2	
10		2		2	4	2		2	
8					4				
6					2				
4					2				
2									

Tabla 4.54 Grafos que requieren replicación en Filtros FIR Paso alto 8 bit

frec\orden	16	14	12	10	8	6	4	2
0,9		R2		R2				
0,8		R2		R2	R5		R1	
0,7	R2				R4			R2
0,6								
0,5	R1				R4			R2
0,4	R3	R2						
0,3		R3	R1	R4				
0,2								
0,1								

Tabla 4.55 Grafos que requieren replicación en Filtros FIR Paso alto 12 bit

frec\orden	16	14	12	10	8	6	4	2
0,9	R2			R4	R2			R2
0,8	R1				R2			R6
0,7	R2							
0,6	R2				R3			R4
0,5							R6	R2
0,4								R7
0,3	R2						R2	R6
0,2							R9	R6
0,1								R6

Tabla 4.56 Grafos que requieren replicación en Filtros FIR Paso alto 16 bit

frec\orden	16	14	12	10	8	6	4	2
0,9		R2						
0,8								
0,7			R1					
0,6								
0,5							R2	
0,4	R2							R2
0,3	R3						R2	
0,2						R1		
0,1								R2

Tabla 4.57 Grafos que requieren replicación en Filtros FIR Paso alto 20 bit

frec\orden	16	14	12	10	8	6	4	2
0,9	R2			R4	R2			R2
0,8	R1				R2			R6
0,7	R2							
0,6	R2				R3			R4
0,5							R6	R2
0,4								R7
0,3	R2						R2	R6
0,2							R9	R6
0,1								R6

Tabla 4.58 Grafos que requieren replicación en Filtros FIR Paso alto 24 bit

frec\orden	16	14	12	10	8	6	4	2
0,9		R2						
0,8								
0,7			R1					
0,6								
0,5							R2	
0,4	R2							R2
0,3	R3						R2	
0,2						R1		
0,1								R2

Tabla 4.59 Área del Registro Check de Filtros FIR DetectError Paso alto 8 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	31834	23052	36224	23052	23052	31834	31834	31834	36224
14	27443	18661	27443	27443	18661	27443	31834	27443	31834
12	23052	18661	23052	23052	14270	23052	27443	23052	27443
10	23052	18661	23052	18661	14270	18661	23052	18661	23052
8	18661	14270	14270	18661	9879	18661	18661	18661	18661
6	14270	14270	9879	14270	9879	14270	14270	14270	14270
4	9879	9879	9879	9879	5489	9879	9879	9879	9879
2	5489	5489	5489	5489	5489	5489	5489	5489	5489

Tabla 4.60 Área del Registro Check de Filtros FIR DetectError Paso alto mayor 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	36224	31834	36224	31834	23052	31834	36224	31834	36224
14	31834	27443	31834	27443	18661	27443	31834	27443	31834
12	27443	23052	27443	23052	14270	23052	27443	23052	27443
10	23052	18661	23052	18661	14270	18661	23052	18661	23052
8	18661	18661	18661	18661	9879	18661	18661	18661	18661
6	14270	14270	14270	14270	9879	14270	14270	14270	14270
4	9879	9879	9879	9879	5489	9879	9879	9879	9879
2	5489	5489	5489	5489	5489	5489	5489	5489	5489

Tabla 4.61 Área del Bloque Multiplicador DetectError Filtros FIR Paso alto 8 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	23644	23464	20790	24675	15351	16459	23564	18857	25124
14	25876	25181	21412	21628	12308	25344	24007	19024	21183
12	22014	22114	18651	10022	14985	17227	17068	15913	17091
10	21602	20950	19772	17896	15568	16499	17593	15847	17836
8	13103	20188	17996	11227	15637	12684	12990	14733	11646
6	10212	15132	15721	9706	13006	15734	14520	15142	13242
4	13788	6014	6393	11037	8655	7405	12617	10139	9953
2	11110	8911	8163	6107	8635	8682	3632	3632	3632

Tabla 4.62 Área total requerida por Filtros FIR DetectError Paso alto 8 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	102839	98269	95594	99479	74535	77585	102759	79983	97480
14	95784	90698	91320	87145	59755	90861	87075	68414	84251
12	82634	78343	79271	66252	59985	73457	70849	62856	70872
10	72935	65444	71105	62390	58119	60993	68926	60341	69169
8	45861	52946	50754	43985	44005	45442	38909	47491	44404
6	52258	57177	57766	51752	43822	57780	49726	50348	55288
4	37259	29485	29864	34508	25287	30876	36088	33610	33424
2	25294	23095	22347	20291	22819	22866	17816	17816	17816

Tabla 4.63 Área del Bloque Multiplicador DetectError Filtros FIR Paso alto 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	36368	33261	36101	36181	20823	33084	40402	29319	40629
14	39341	34365	31900	29375	21346	26086	35772	30972	36224
12	34585	27586	31045	26491	23960	30753	30107	27200	30563
10	30856	32349	27805	27137	18042	32476	28258	27785	31262
8	24722	24429	27742	23368	27220	23894	22666	25015	27240
6	19210	23687	24007	18791	18575	22792	23375	20840	20191
4	18155	14822	13748	14164	14034	16067	18352	16313	21063
2	14190	14097	14104	13346	13545	21924	3632	13954	8210

Tabla 4.64 Área total requerida por Filtros FIR DetectError Paso alto 12 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	151787	139898	151521	142819	103059	139722	155822	135957	156048
14	141083	127325	133641	122335	96742	119045	137513	123932	137966
12	122648	106867	119108	105773	83230	110034	118170	106481	118626
10	105241	95504	102190	90292	74864	95631	102643	90940	105646
8	85429	85136	88449	84075	67915	84600	83373	85721	87947
6	66239	70716	71035	65819	56822	69821	70403	67869	67220
4	51506	48173	47098	47514	36155	49417	51702	49663	54413
2	33863	33770	33776	33018	33217	41597	23305	33627	27882

Tabla 4.65 Área del Bloque Multiplicador DetectError Filtros FIR Paso alto 16 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	50165	48269	56452	50405	31900	44358	58109	39681	57041
14	51938	45821	54456	47840	31464	43982	44238	32599	57487
12	46347	45392	44647	37898	32293	32911	35293	36234	40828
10	38057	35832	33454	40762	27047	35839	35010	38995	44125
8	35237	33417	32891	30776	27955	30796	31814	34545	35885
6	24908	31108	28793	24133	31378	26538	28062	26950	28727
4	28271	19203	20537	17906	27689	21691	24968	21705	22347
2	14284	19909	14104	19133	19133	23544	6177	16822	14071

Tabla 4.66 Área total requerida por Filtros FIR DetectError Paso alto 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	165585	154907	171872	157043	114135	150995	173528	146318	172461
14	153680	138781	156198	140800	106861	136941	145979	125558	159228
12	134410	124673	132710	117179	91562	112193	123356	115516	128891
10	112442	98987	107839	103917	83869	98994	109395	102150	118510
8	95943	94124	93598	91483	68650	91503	92520	95251	96592
6	71937	78137	75822	71162	69625	73567	75090	73979	75755
4	61622	52554	53888	51256	49810	55042	58318	55055	55697
2	33956	39581	33776	38806	38806	43217	25849	36494	33743

Tabla 4.67 Área del Bloque Multiplicador DetectError Filtros FIR Paso alto 20 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	69598	63621	66491	59805	41154	59526	66847	56921	70776
14	72516	49450	66748	59566	37206	50445	58262	48017	56702
12	58887	53778	53472	49666	37878	52078	48392	47022	56436
10	46127	39494	50205	44301	37226	45289	47448	42787	56020
8	41813	43227	40323	41989	34495	37548	41500	45412	45978
6	37944	38423	39275	30061	39421	30896	36703	39431	37163
4	45236	32083	27283	44497	34734	24166	39328	28657	25540
2	19792	19909	34525	27559	21322	22802	19652	28767	28358

Tabla 4.68 Área total requerida por Filtros FIR DetectError Paso alto 20 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	185018	170258	181911	166443	123389	166164	182267	163559	186195
14	174257	142410	168489	152525	112602	143404	160003	140976	158443
12	146950	133059	141535	128948	97148	131360	136456	126303	144499
10	120512	102649	124590	107456	94047	108444	121833	105943	130405
8	102520	103933	101029	102696	75190	98255	102207	106119	106684
6	84973	85452	86303	77089	77668	77924	83732	86460	84191
4	78586	65434	60634	77848	56855	57517	72679	62007	58891
2	39464	39581	54197	47232	40995	42475	39325	48439	48030

Tabla 4.69 Área del Bloque Multiplicador DetectError Filtros FIR Paso alto 24 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	82046	75183	100950	76221	51599	70769	83602	73214	87750
14	86097	69153	71697	69326	51692	67559	82551	67849	76787
12	66265	67400	62796	61033	45096	57454	66182	57959	68224
10	65873	56216	63168	55255	45299	55411	63108	53279	73054
8	51253	54077	53948	49457	65600	52271	49031	60012	51659
6	45625	49670	40007	41058	46832	46636	45562	41909	43466
4	70170	46383	44421	41923	39218	35459	47148	60790	55255
2	31617	19909	24183	27652	24775	33141	28467	24921	27054

Tabla 4.70 Área total requerida por Filtros FIR DetectError Paso alto 24 bit

frec\orden	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	197465	181821	216369	182859	133834	177407	199022	179852	203170
14	187838	162112	173438	162285	127088	160519	184293	160808	178528
12	154328	146681	150859	140314	104366	136735	154245	137241	156288
10	140258	119371	137553	118410	102120	118566	137493	116434	147439
8	111960	114784	114654	110164	106295	112978	109738	120718	112366
6	92654	96698	87035	88086	85079	93665	92590	88938	90495
4	103521	79734	77771	75273	61339	68810	80499	94140	88605
2	51290	39581	43855	47325	44447	52813	48140	44594	46726

Tabla 4.71 Área del Comparador Filtros FIR DMR Paso alto de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	16466	12075	18661	12075	12075	16466	16466	16466	18661
14	14270	9879	14270	14270	9879	14270	16466	14270	16466
12	12075	9879	12075	12075	7684	12075	14270	12075	14270
10	12075	9879	12075	9879	7684	9879	12075	9879	12075
8	9879	7684	7684	9879	5489	9879	9879	9879	9879
6	7684	7684	5489	7684	5489	7684	7684	7684	7684
4	5489	5489	5489	5489	3293	5489	5489	5489	5489
2	3293	3293	3293	3293	3293	3293	3293	3293	3293

Tabla 4.72 Área del Comparador Filtros FIR DMR Paso alto mayor de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	18661	16466	18661	16466	12075	16466	18661	16466	18661
14	16466	14270	16466	14270	9879	14270	16466	14270	16466
12	14270	12075	14270	12075	7684	12075	14270	12075	14270
10	12075	9879	12075	9879	7684	9879	12075	9879	12075
8	9879	9879	9879	9879	5489	9879	9879	9879	9879
6	7684	7684	7684	7684	5489	7684	7684	7684	7684
4	5489	5489	5489	5489	3293	5489	5489	5489	5489
2	3293	3293	3293	3293	3293	3293	3293	3293	3293

Tabla 4.73 Área del Bloque Multiplicador de Filtros FIR DMR Paso alto de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	50249	37715	47128	28527	30703	49350	37189	46929	47288
14	42365	38047	42279	45718	24615	43257	42824	45086	46663
12	34182	31827	34136	34455	29971	20045	37302	44228	44028
10	35672	31694	29911	32998	31135	35792	39544	36670	38812
8	26485	30284	29039	31468	26012	19413	31441	30264	20424
6	23291	29465	25979	25367	22493	22453	26797	31321	26205
4	19905	20278	25234	14809	17311	22074	12787	7637	27576
2	7265	7265	7265	17364	12454	12215	11935	17823	22220

Tabla 4.74 Área Total de Filtros FIR DMR Paso alto de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	139070	110915	144984	101728	101961	140620	128459	138199	145144
14	119704	97317	119617	125505	81943	123044	129197	124873	133036
12	100038	88649	99992	102759	82654	88349	112193	112532	118919
10	99080	86067	93319	87371	81370	90165	102952	91044	102220
8	78410	73174	71930	83393	62317	71338	83366	82189	72349
6	63734	69908	57387	65809	56349	62896	67240	71764	66648
4	48865	49237	54194	43769	37236	51034	41746	36597	56536
2	24742	24742	24742	34841	29931	29691	29412	35300	39697

Tabla 4.75 Área del Bloque Multiplicador de Filtros FIR DMR Paso alto de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	72735	66521	72203	72363	41647	66169	75995	58638	81257
14	78683	68730	63800	58751	42691	47202	71544	61944	72449
12	63740	55172	57001	52983	47920	61505	60214	54400	61126
10	57321	55491	55611	54274	36085	60560	51699	55571	62523
8	49444	48858	55484	46736	36457	42970	45332	50029	54480
6	38420	47375	48013	37582	32758	45585	46749	41680	40382
4	36311	29645	27496	28328	28068	32133	36703	32625	32459
2	18289	18102	22779	22021	17457	28940	7265	23518	12028

Tabla 4.76 Área Total de Filtros FIR DMR Paso alto de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	170591	157791	170059	163632	112905	157439	173851	149908	179113
14	165056	148517	150174	138538	109306	126989	157918	141731	158822
12	138631	123476	131892	121287	100604	129809	135105	122704	136016
10	120728	109864	119019	108647	86320	114934	115107	109944	125931
8	101369	100783	107409	98661	72762	94896	97257	101954	106405
6	78862	87817	88456	78024	66614	86027	87192	82122	80825
4	65271	58605	56456	57287	47993	61093	65663	61585	61419
2	35765	35579	40256	39498	34934	46417	24742	40995	29505

Tabla 4.77 Área del Bloque Multiplicador de Filtros FIR DMR Paso alto de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	100331	92148	106405	95441	63800	88715	109718	79361	114082
14	103877	91642	95454	95681	62929	87963	84085	65197	104116
12	92693	90784	89294	75795	64585	65823	70586	72469	81656
10	76115	71664	66907	76707	54094	71677	70021	73321	88249
8	63820	66834	65783	61552	55910	61592	63627	69089	71770
6	49816	62217	57587	48266	38939	53076	56123	53901	57454
4	41753	38407	41074	35812	35193	43383	49936	43410	44694
2	18475	24216	22779	23418	23325	30955	12354	23977	28141

Tabla 4.78 Área Total de Filtros FIR DMR Paso alto de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	198187	183418	204261	186711	135058	179985	207574	170631	211938
14	190250	171429	181828	175468	129543	167750	170458	144984	190490
12	167584	159088	164184	144100	117269	134127	145477	140773	156547
10	139523	126037	130315	131080	104329	126051	133429	127694	151657
8	115745	118759	117708	113477	92214	113517	115552	121014	123696
6	90259	102659	98029	88708	72795	93518	96565	94343	97896
4	70713	67366	70034	64772	55118	72343	78896	72369	73653
2	35952	41693	40256	40895	40802	48432	29831	41454	45618

Tabla 4.79 Área del Bloque Multiplicador de Filtros FIR DMR Paso alto de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	139197	127241	132983	119611	82308	112565	133695	113843	141552
14	137919	98901	119730	112326	74412	100890	116524	96033	113404
12	117775	107556	106944	99333	75755	98422	96785	94044	112871
10	92254	78989	100411	88602	74451	90578	94896	85575	112040
8	83626	86453	80645	83978	58970	75097	83000	90824	91955
6	75888	76846	78550	60121	51160	61791	73407	72209	74325
4	53848	64166	49444	56043	42471	48333	49364	57314	51080
2	24216	24216	42565	34901	25746	30516	24030	36550	36211

Tabla 4.80 Área Total de Filtros FIR DMR Paso alto de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	237053	218511	230839	210880	153567	203835	231551	205112	239408
14	224293	178688	206104	192113	141026	180677	202897	175820	199777
12	192665	175860	181834	167637	128439	166726	171676	162348	187762
10	155662	133362	163819	142975	124687	144951	158303	139948	175448
8	135551	138378	132570	135903	95275	127022	134925	142749	143880
6	116331	117289	118992	100564	85016	102234	113849	112652	114767
4	82807	93126	78403	85003	62397	77292	78323	86274	80040
2	41693	41693	60042	52377	43223	47993	41507	54027	53688

Tabla 4.81 Área del Bloque Multiplicador de Filtros FIR DMR Paso alto de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	164091	150367	185959	137520	97157	141538	167205	135997	175501
14	165289	138305	143394	138651	97177	135118	153387	122345	153573
12	132530	134799	125592	122066	90192	114907	132364	115918	136449
10	125538	112432	126337	110510	90598	110822	126217	106558	146109
8	102506	108155	107895	98914	80486	104542	98062	120023	103318
6	91250	93412	80013	82116	55418	93272	91123	83819	86932
4	79368	55238	57008	72196	44061	70919	63627	76095	69535
2	40170	24216	28700	36065	32652	37914	36717	29392	37668

Tabla 4.82 Área Total de Filtros FIR DMR Paso alto de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	261947	241636	283815	228790	168416	232808	265061	227266	273357
14	251662	218092	229768	218438	163792	214905	239760	202132	239947
12	207421	203103	200482	190370	142876	183211	207255	184223	211339
10	188946	166806	189745	164883	140833	165196	189625	160931	209517
8	154431	160080	159820	150839	116790	156467	149987	171948	155243
6	131692	133854	120456	122558	89274	133715	131566	124261	127375
4	108328	84198	85967	101156	63987	99878	92587	105054	98495
2	57647	41693	46177	53542	50129	55391	54194	46869	55145

Tabla 4.83 Ahorro de Área usando Filtros FIR Paso bajo DetectError de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	-1%	3%	2%	3%	2%	-1%	-2%	0%	-6%
15	3%	1%	0%	0%	2%	-2%	-1%	-6%	-2%
14	1%	2%	1%	4%	3%	4%	-3%	2%	-3%
13	3%	3%	2%	0%	0%	0%	-2%	-1%	0%
12	5%	7%	5%	3%	6%	1%	2%	1%	1%
11	7%	3%	-1%	1%	0%	7%	0%	4%	-1%
10	2%	5%	2%	4%	6%	5%	2%	1%	1%
9	4%	0%	7%	-1%	4%	6%	4%	3%	-1%
8	6%	7%	7%	2%	8%	5%	5%	-3%	1%
7	10%	2%	1%	1%	7%	6%	7%	-1%	3%
6	7%	-1%	13%	9%	4%	-1%	6%	1%	5%
5	6%	9%	8%	6%	2%	10%	4%	7%	7%
4	9%	5%	13%	10%	13%	6%	1%	1%	11%
3	1%	1%	-8%	-3%	4%	10%	5%	10%	10%
2	9%	9%	16%	15%	15%	9%	11%	10%	10%

Tabla 4.84 Ahorro de Área usando Filtros FIR Paso bajo DetectError de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	10%	9%	4%	7%	10%	9%	6%	8%	3%
15	2%	9%	8%	7%	9%	7%	6%	6%	5%
14	9%	6%	5%	10%	9%	10%	8%	8%	6%
13	4%	9%	9%	9%	7%	4%	6%	5%	7%
12	6%	12%	6%	10%	10%	9%	8%	8%	4%
11	9%	9%	8%	6%	8%	10%	11%	7%	6%
10	12%	12%	10%	12%	14%	8%	9%	8%	6%
9	11%	13%	10%	2%	10%	12%	11%	8%	9%
8	13%	13%	12%	13%	12%	13%	10%	10%	7%
7	4%	3%	14%	9%	13%	9%	5%	12%	12%
6	16%	14%	14%	15%	8%	11%	10%	17%	16%
5	13%	13%	16%	11%	3%	5%	12%	8%	12%
4	13%	20%	20%	13%	8%	14%	2%	15%	18%
3	2%	1%	1%	5%	6%	5%	6%	15%	16%
2	21%	21%	20%	9%	9%	8%	9%	9%	8%

Tabla 4.85 Ahorro de Área usando Filtros FIR Paso bajo DetectError de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	14%	16%	11%	14%	15%	14%	11%	16%	11%
15	10%	15%	16%	12%	15%	12%	15%	13%	10%
14	13%	16%	13%	16%	16%	15%	12%	15%	13%
13	13%	14%	15%	14%	15%	14%	15%	13%	11%
12	16%	17%	17%	17%	18%	18%	18%	14%	14%
11	14%	12%	14%	14%	16%	16%	16%	11%	15%
10	13%	21%	19%	18%	17%	18%	17%	17%	14%
9	16%	17%	16%	16%	17%	17%	16%	12%	17%
8	17%	16%	18%	19%	21%	19%	15%	15%	20%
7	16%	19%	18%	13%	5%	19%	15%	13%	16%
6	20%	18%	18%	22%	21%	14%	17%	19%	21%
5	3%	2%	20%	19%	17%	5%	16%	20%	18%
4	23%	23%	22%	25%	8%	22%	16%	21%	22%
3	2%	2%	2%	19%	5%	6%	15%	5%	5%
2	20%	8%	25%	9%	20%	8%	8%	9%	8%

Tabla 4.86 Ahorro de Área usando Filtros FIR Paso bajo DetectError de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	17%	17%	19%	18%	25%	17%	19%	20%	17%
15	20%	23%	19%	20%	15%	23%	20%	15%	17%
14	19%	20%	17%	19%	20%	18%	18%	20%	17%
13	16%	19%	21%	15%	20%	19%	16%	21%	16%
12	21%	23%	21%	20%	23%	20%	18%	18%	18%
11	18%	18%	17%	19%	20%	19%	21%	17%	19%
10	22%	23%	21%	20%	22%	23%	19%	21%	22%
9	20%	21%	21%	13%	20%	21%	20%	20%	19%
8	24%	25%	22%	23%	14%	22%	14%	21%	23%
7	22%	3%	23%	21%	21%	16%	20%	19%	21%
6	23%	24%	25%	24%	25%	24%	20%	23%	24%
5	11%	2%	5%	23%	16%	5%	24%	19%	5%
4	26%	28%	8%	9%	7%	6%	7%	27%	28%
3	12%	2%	2%	5%	5%	4%	6%	5%	5%
2	9%	9%	8%	21%	8%	7%	18%	14%	8%

Tabla 4.87 Ahorro de Área usando Filtros FIR Paso bajo DetectError de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	22%	20%	23%	24%	28%	22%	22%	23%	19%
15	23%	26%	21%	24%	19%	25%	25%	23%	22%
14	23%	25%	23%	25%	26%	24%	23%	22%	22%
13	23%	24%	23%	23%	24%	23%	25%	22%	20%
12	26%	22%	24%	26%	25%	24%	23%	24%	22%
11	24%	22%	22%	24%	23%	24%	20%	22%	25%
10	26%	27%	26%	28%	25%	27%	24%	25%	24%
9	24%	23%	26%	22%	24%	25%	21%	24%	25%
8	25%	25%	27%	28%	9%	27%	26%	25%	24%
7	26%	3%	26%	21%	26%	12%	26%	21%	24%
6	23%	29%	31%	26%	5%	28%	26%	28%	27%
5	27%	2%	5%	3%	3%	5%	5%	6%	28%
4	5%	5%	9%	5%	16%	7%	19%	30%	7%
3	12%	2%	2%	5%	5%	6%	15%	6%	5%
2	8%	9%	7%	8%	7%	7%	16%	8%	17%

Tabla 4.88 Ahorro de Área usando Filtros FIR Paso alto DetectError de 8 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	7%	7%	4%	1%	4%	7%	1%	6%	4%
14	7%	11%	4%	6%	4%	7%	5%	5%	4%
12	6%	8%	6%	6%	10%	-1%	5%	10%	7%
10	7%	8%	1%	9%	11%	10%	9%	8%	6%
8	6%	12%	11%	8%	14%	1%	8%	8%	2%
6	8%	12%	15%	9%	4%	7%	3%	6%	10%
4	11%	12%	15%	7%	17%	13%	5%	-8%	17%
2	6%	6%	6%	19%	5%	13%	5%	19%	22%

Tabla 4.89 Ahorro de Área usando Filtros FIR Paso alto DetectError de 12 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	11%	11%	11%	13%	9%	11%	10%	9%	13%
14	15%	14%	11%	12%	11%	6%	13%	13%	13%
12	12%	13%	10%	13%	17%	15%	13%	13%	13%
10	13%	13%	14%	17%	13%	17%	11%	17%	16%
8	16%	16%	18%	15%	7%	11%	14%	16%	17%
6	16%	19%	20%	16%	15%	19%	19%	17%	17%
4	21%	18%	17%	17%	25%	19%	21%	19%	11%
2	5%	5%	16%	16%	5%	10%	6%	18%	6%

Tabla 4.90 Ahorro de Área usando Filtros FIR Paso alto DetectError de 16 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	16%	16%	16%	16%	15%	16%	16%	14%	19%
14	19%	19%	14%	20%	18%	18%	14%	13%	16%
12	20%	22%	19%	19%	22%	16%	15%	18%	18%
10	19%	21%	17%	21%	20%	21%	18%	20%	22%
8	17%	21%	20%	19%	26%	19%	20%	21%	22%
6	20%	24%	23%	20%	4%	21%	22%	22%	23%
4	13%	22%	23%	21%	10%	24%	26%	24%	24%
2	6%	5%	16%	5%	5%	11%	13%	12%	26%

Tabla 4.91 Ahorro de Área usando Filtros FIR Paso alto DetectError de 20 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	22%	22%	21%	21%	20%	18%	21%	20%	22%
14	22%	20%	18%	21%	20%	21%	21%	20%	21%
12	24%	24%	22%	23%	24%	21%	21%	22%	23%
10	23%	23%	24%	25%	25%	25%	23%	24%	26%
8	24%	25%	24%	24%	21%	23%	24%	26%	26%
6	27%	27%	27%	23%	9%	24%	26%	23%	27%
4	5%	30%	23%	8%	9%	26%	7%	28%	26%
2	5%	5%	10%	10%	5%	11%	5%	10%	11%

Tabla 4.92 Ahorro de Área usando Filtros FIR Paso alto DetectError de 24 bit

orden\frec	0,9	0,8	0,7	0,6	0,5	0,4	0,3	0,2	0,1
16	25%	25%	24%	20%	21%	24%	25%	21%	26%
14	25%	26%	25%	26%	22%	25%	23%	20%	26%
12	26%	28%	25%	26%	27%	25%	26%	26%	26%
10	26%	28%	28%	28%	27%	28%	27%	28%	30%
8	28%	28%	28%	27%	9%	28%	27%	30%	28%
6	30%	28%	28%	28%	5%	30%	30%	28%	29%
4	4%	5%	10%	26%	4%	31%	13%	10%	10%
2	11%	5%	5%	12%	11%	5%	11%	5%	15%

Anexo A-10 Trabajo bibliográfico previo.

10.1 Contenido energético de los Rayos Cósmicos.

En la página web de Seutest (http://www.seutest.com/) se encuentran links y soporte técnico del standard JESD89 de JEDEC "Measurement and Reporting of Alpha Particles and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices." [16], con el que se ha construido la figura 10.1, donde se visualiza el flujo de partículas cósmicas según su energía.

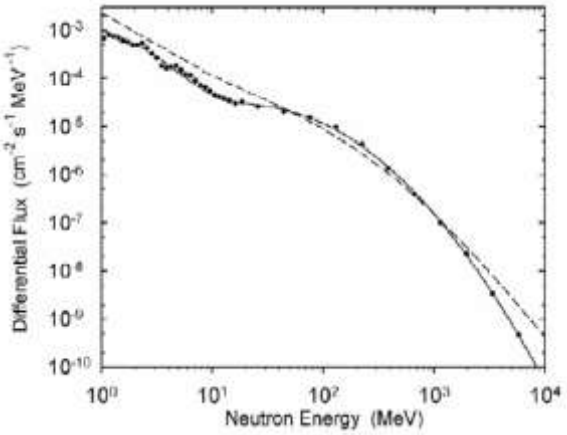


Figura 10.1 Flujo de partículas cósmicas según su contenido energético

En el menú online de http://www.seutest.com/cgi-bin/FluxCalculator.cgi recogido en la Figura 10.2, se puede calcular el flujo incidente sobre cualquier punto de la geografía terrestre sin más que introducir sus coordenadas y su elevación sobre el nivel del mar.

El flujo de rayos cósmicos aumenta unas 2,4 veces cada 1000 metros de forma que cualquier avión comercial recibe un flujo más de 300 veces superior al experimentado al nivel del mar (13 neutrones de energía superior a 10MeV/cm^2 hora en la ciudad de Nueva York) [5].

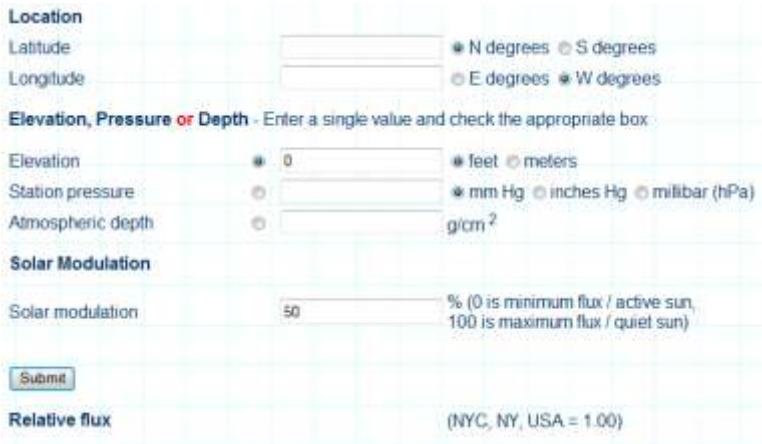


Figura 10.2 Flujo de partículas cósmicas según coordenadas

10.2 Técnicas para reducir los Soft Errors.

En este punto se recogen las soluciones descritas para reducir la magnitud de Soft Errors tienen en cuenta las siguientes estrategias [17]:

10.2.1 Modificación constructiva de los componentes de los circuitos.

10.2.2 Modificación de diseño de los circuitos.

10.2.3 Modificación del software de los circuitos.

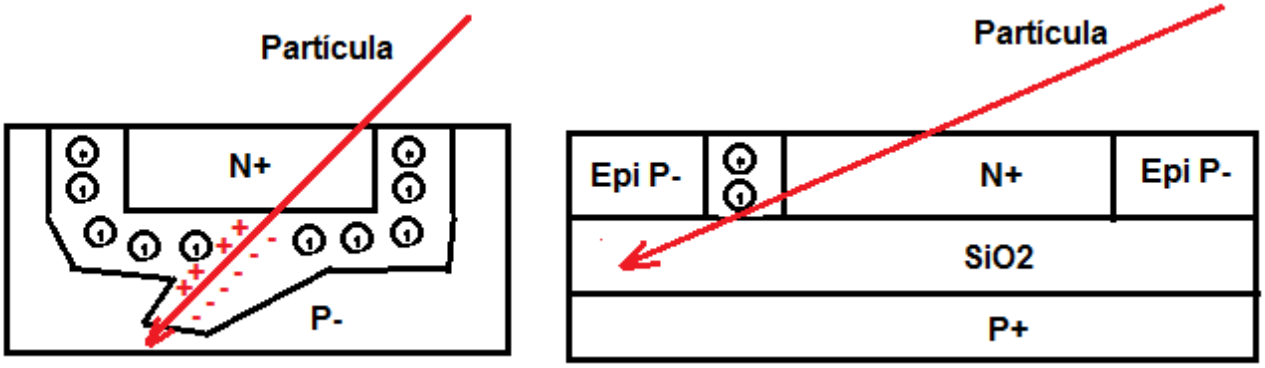
10.2.1 Modificacion constructiva de los componentes de los circuitos.

Silicon on insulator *(SOI)*

Es una tecnología de fabricación microelectrónica en la que, como se indica en la figura 10.3, se sustituye el sustrato tradicional de fabricación de obleas de silicio monocristalino, por un sándwich de capas de semiconductor-aislante-semiconductor [18].

Esta técnica reduce las capacidades parásitas de los circuitos fabricados, reduce el riesgo de latch-up en los circuitos lógicos CMOS, y mejora la escalabilidad de los circuitos integrados.

El aislante empleado suele ser típicamente dióxido de silicio o, en aplicaciones en las que se busca resistencia frente a la radiación, zafiro.



Diseño standard **Silicon On Insulator**

Figura 10.3 Diferencia diseño estándar / SOI

Las láminas de semiconductor suelen ser de silicio, aunque se buscan nuevas alternativas para mejorar las prestaciones de los dispositivos introduciendo nuevos materiales semiconductores como silicio tenso y aleaciones de silicio/germanio.

Dependiendo del espesor de la lámina de silicio sobre el aislante se distinguen dos tipos de tecnología SOI.

FD-SOI (Fully-Depleted SOI):

Cuando la lámina de silicio se encuentra completa-mente deplexionada de portadores móviles (electrones o huecos).

PD-SOI (Partially-Depleted SOI):

Cuando está parcialmente deplexionada.

La tecnología FD-SOI es muy prometedora para la miniaturización de los dispositivos electrónicos, mientras que la PD-SOI muestra sus ventajas en la fabricación de transistores que deban operar a altas frecuencias (como en microprocesadores) así como en la fabricación de memorias de un solo transistor (1T-DRAM).

El uso de SOI tiene la ventaja de no requerir apenas cambios en el proceso de fabricación de los circuitos integrados más allá de utilizar obleas SOI distintas de las tradicionales.

Su principal inconveniente es el coste: las obleas SOI son significativamente mucho más caras que las ordinarias.

10.2.2 Modificación del diseño de los circuitos.

Las técnicas de reducción de Soft Error a nivel de diseño de circuito más empleadas son:

Redundancia Triple Modular de células de Memoria con Voto ponderado (TMR).

La solución implica la triplicación del número de células de memoria y la implementación de un algoritmo de Voto ponderado para desechar el almacenamiento del valor erróneo.

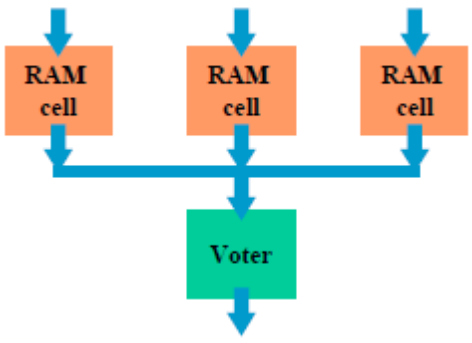


Figura 10.4 Redundancia triple modular (TMR)

Uso de Células de Memorias con Gate Resistors.

Esta solución utiliza Gate Resistors para proteger los datos de las células de memoria de Soft Errors [19].

La alta resistencia del transistor protege los datos almacenados en la célula al distinguir un pulso ocasional de corriente, originado por una partícula cargada, de las señales estándar de escritura, de mayor tiempo de duración.

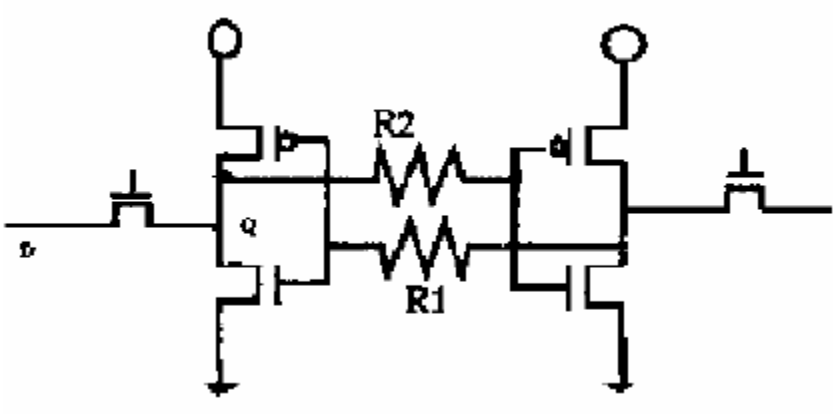


Figura 10.5 SRAM cell based on gate resistor

Uso de Células de Memoria con capacidad de recuperación de datos.

La idea básica es utilizar células de memoria con un sistema que permite guardar los datos en dos localizaciones distintas de forma que ante un Soft Error el dato corrupto pueda ser eliminado recuperando el dato correcto.

Estas células de memoria incorporan un número más elevado de transistores en su diseño.

A continuación se describe su funcionamiento.

Veamos en primer lugar en la figura 10.6, el esquema constructivo de una célula de memoria convencional con 6 transistores, y que es sensible a un Soft Error [20]

Ante una entrada “0”, los transistores P2 and N1 se encuentran ON y los transistores P1 and N2 son OFF.

Si una partícula cargada incide sobre el hilo de P1 o N2, el transistor cuya puerta está conectada a ese hilo comenzará a conducir.

De esta forma, si P1 fuese alcanzado por esa particular, N2 se volvería ON y el nodo A contendrá “1” y no el “0” inicial.

En ese momento el Soft Error ha invertido el contenido de la memoria.

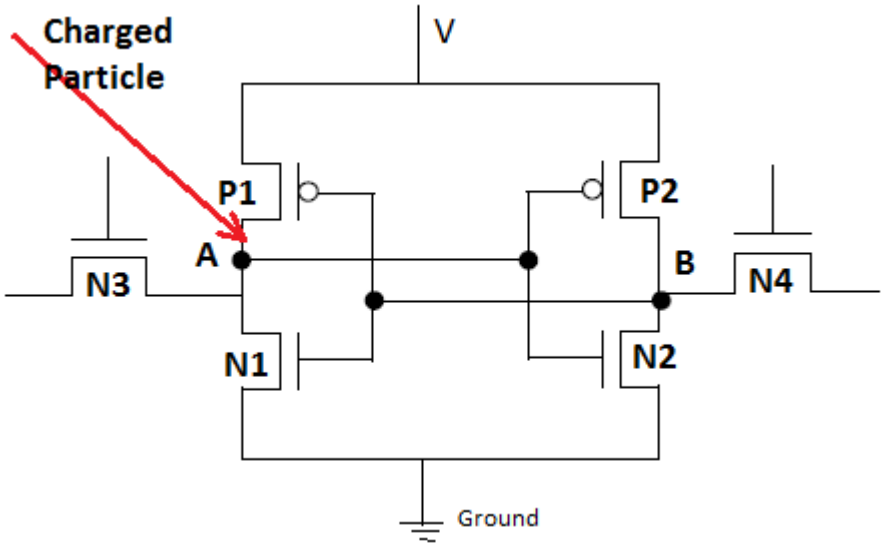


Figura 10.6 Célula de Memoria Soft Error sensible

En la figura 10.7 se representa la estructura de una de las células de memoria inmune a Soft Errors existentes, la llamada NASA Memory Cell I, también conocida como Célula de Memoria Whitaker [21], que está constituida por 16 transistores.

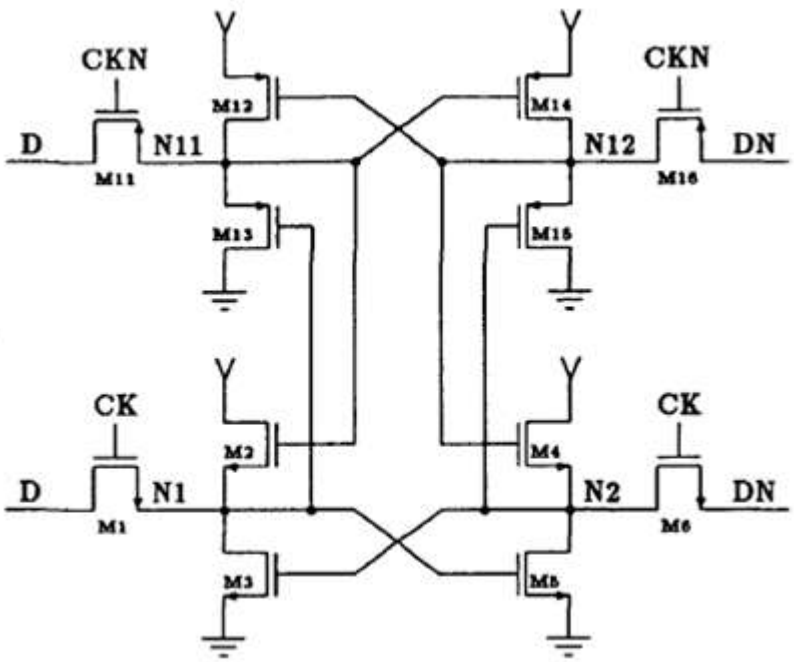


Figura 10.7 NASA Memory Cell I

Esta célula de memoria está organizada en dos partes bien diferenciadas. La parte superior está compuesta exclusivamente por p-channel transistores. La inferior solo por n-channel transistores. Los transistores M2 y M4 se dimensionan para tener menos potencia que M3 y M5, mientras M13 y M15 son de menor potencia que M12 y M14

Los nodos N1 y N2 pueden almacenar “0” y no pueden ser modificados mientras que los nodos N11 y N12 solo pueden almacenar “1” que tampoco pueden ser modificados.

Si N11 tuviese almacenado un “0” y una partícula cargada incide en el nodo y cambia a 1, M14 se activa pero N12 continúa como “1”. M2 se activa pero como tiene menor potencia que M3 no puede sobrecargar a N1 permaneciendo M13 activo y reescribiendo N11 como “0”.

Si N1 tuviese almacenado un “1” y una partícula cargada incide en el nodo y cambia a “0”, M5 se desactiva haciendo que N2 pase a “0”. M13 se activa pero como tiene menor potencia que M12 no puede sobrecargar a N11 permaneciendo M2 activo y reescribiendo N1 como “1”.

En ambas situaciones la célula de memoria no ha sufrido cambio, no se ha producido Soft Error.

10.2.3 Modificación del software de los circuitos.

Bloques lógicos de Detección y Corrección de Errores.

A este tipo pertenecen los Bloques Codificadores Hamming EDAC [22], también llamados Hamming (n,k).

Los coeficientes tienen el siguiente significado:

n representa el número de bits de la palabra codificada.

k es el número de bits de la palabra inicial, tal y como se indica en la figura 10.8.

La principal característica de los Hamming códigos es que pueden DETECTAR errores dobles y CORREGIR errores simples (SEC-DED).

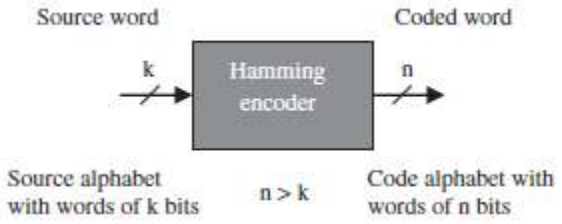


Figura 10.8 Bloque codificador Hamming EDAC

La codificación Hamming se lleva a cabo mediante el uso de la matriz G, llamada **Matriz generadora de código**, con la que se calcula la paridad de los bits de cada palabra inicial a codificar.

En la figura 10.9 se indica una posible matriz G para una codificación Hamming (12,8).

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Figura 10.9 Code generator matrix

El procedimiento a seguir para obtener la palabra codificada es multiplicar (en modo binario) la palabra inicial por la matriz G:

$$c = x * G$$

Esta palabra codificada contiene sus datos en los primeros k bits y su paridad en los últimos (n– k) bits.

La detección de errores se realiza mediante el concurso de la matriz H^T, **Matriz de Chequeo de Paridad.**

Esta matriz, que es ortogonal con respecto a la Matriz generadora de código descrita con anterioridad, nos permite el cálculo del **error syndrome**, s, de acuerdo con la siguiente expresión:

$$s = c * H^T$$

Cuando la palabra codificada no contiene ningún error en ninguno de sus bits, el error síndrome, s, es nulo, mientras que ante cualquier error en la palabra codificada el resultado del cálculo es distinto de cero.

En la figura 10.10 se indica la disposición de elementos una posible matriz que cumple dichas condiciones.

$$H^T = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 10.10 Check parity matrix

10.3 Filtros Digitales.

10.3.1 Expresión matemática de los Filtros Digitales descritos:

Expresión matemática de un filtro IIR.

La salida de los Filtros IIR depende de las entradas actuales y pasadas, y además de las salidas en instantes anteriores. Esto se consigue mediante el uso de realimentación de la salida.

Para un Filtro IIR de forma directa Tipo I

$$\boldsymbol{y_n = b_0 x_n + b_1 x_{n-1} + \cdots + b_N x_{n-N} - a_1 y_{n-1} - a_2 y_{n-2} - \cdots - a_M y_{n-M}}$$

$$\boldsymbol{y_n = \sum_{k=0}^{N} b_k x_{(n-k)} - \sum_{k=1}^{M} a_k y_{(n-k)}}$$

Donde **a** y **b** son los coeficientes del filtro. M y N son los términos que determinan la cantidad de polos y ceros en la función de transferencia.

El orden es el máximo entre los valores de M y N. Normalmente se asume que N y M son iguales.

Aplicando la transformada Z a la expresión anterior:

$$\boldsymbol{H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{M} a_k z^{-k}}}$$

Expresión matemática de un filtro FIR Tipo directo.

La respuesta del filtro es:

$$\boldsymbol{y_n = b_0 x_n + b_1 x_{n-1} + \cdots + b_M x_{n-M}}$$

$$\boldsymbol{y_n = \sum_{k=0}^{M} b_k x_{(n-k)}} \quad \textbf{(1)}$$

Siendo M es el orden del filtro. El número de coeficientes del filtro es $M+1$ Los coeficientes son b_k.

La salida también puede expresarse como la convolución de la señal de entrada x_n con la respuesta al impulso h_n:

$$\boldsymbol{y_n = \sum_{k=-\infty}^{+\infty} h_k\ x_{n-k}} \qquad \text{con } h_k = \begin{cases} \boldsymbol{0} & \boldsymbol{k < 0} \\ \boldsymbol{h_k} & \text{para } \boldsymbol{0 \le k < M} \\ \boldsymbol{0} & \boldsymbol{k \ge M} \end{cases}$$

luego la expresión puede reescribirse como:

$$\boldsymbol{y_n = \sum_{k=0}^{M} h_k\ x_{n-k}} \quad \textbf{(2)}$$

Identificando las expresiones (1) y (2), observamos que $b_k=\{h(_k)\}$, es decir, los coeficientes del filtro corresponden a la respuesta impulsional.

Aplicando la transformada Z a la expresión anterior:

$$\boldsymbol{H(z) = \sum_{k=0}^{M} h_k z^{-k} = h_0 + h_1\ z^{-1} + \ldots + h_M\ z^{-M}}$$

Expresión matemática de un filtro FIR Tipo Cascada.

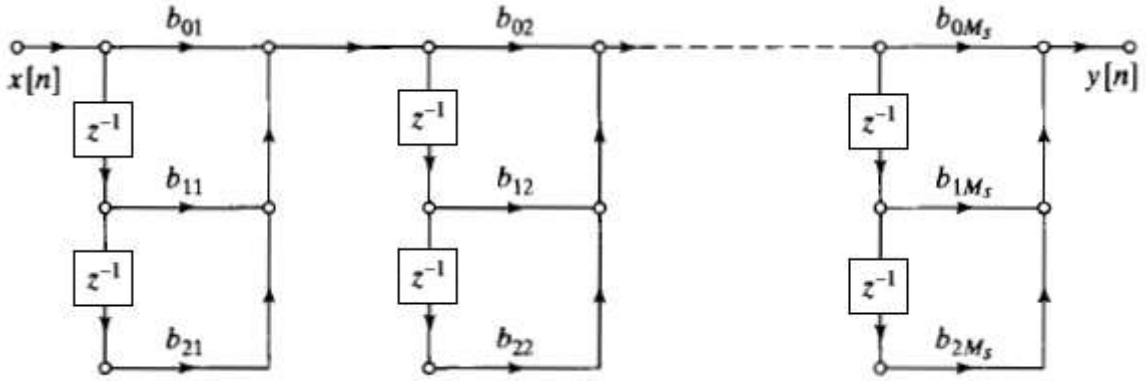


Figura 10.11 FIR Tipo cascada

Para este tipo de filtro, aplicando la transformada Z obtenemos:

$$\boldsymbol{H(z) = \sum_{n=0}^{M} h_n \; z^{-n} \; = \; \prod_{k=1}^{M} (b_{0k} \; + \; b_{1k}\, z^{-1} \; + \; b_{2k}\, z^{-2})}$$

10.3.2 Respuestas de frecuencia de Filtros FIR Ideales.

A continuación se presenta la respuesta en frecuencia para cada uno de estos tipos de filtro.

Respuesta de frecuencia para FIR Tipo I:

Teniendo en cuenta que es un filtro de longitud impar y respuesta impulsional simétrica alrededor del punto central, tenemos que:

$$\boldsymbol{H(z) = \sum_{n=0}^{M} h_n\, z^{-n} \; = \; h_0 \; + \; h_1 \; z^{-1} + \; \ldots \; + \; h_M \; z^{-M}}$$

con

$$\boldsymbol{h_n = \; h_{M-n} \quad para \;\; 0 \leq n \leq M}$$

Realizando el cambio de variable $z = e^{j\omega}$

$$\boldsymbol{H(e^{j\omega}) = \sum_{n=0}^{M} h_n\, e^{-j\omega n}}$$

Aprovechando la propiedad de simetría de los coeficientes, la respuesta en frecuencia resulta ser:

$$\boldsymbol{H(e^{j\omega}) = e^{-j\omega\frac{M}{2}}\left(\sum_{k=0}^{M/2} a_k \cos(\omega k)\right)}$$

Con lo que:

$$\boldsymbol{H(\omega) = A(\omega)\, e^{j\Phi(\omega)}} \qquad \text{siendo} \qquad \boldsymbol{A(\omega) = \sum_{k=0}^{M/2} a_k \cos(\omega k)}$$

$$\boldsymbol{\Phi(\omega) = -\omega\frac{M}{2}}$$

Como ***A(ω)*** $\in$ R, toda la información de la fase se encuentra en **Φ(ω)** que es lineal en **ω**. Por tanto, el retardo de grupo es lineal:

$$\boldsymbol{\tau(\omega) = -\frac{d\Phi(\omega)}{d\omega} = \frac{M}{2}}$$

Esto significa que, al hacer pasar una señal por un filtro FIR de estas características, el retardo es el mismo para todos los armónicos que componen la señal y ésta no se distorsiona.

La magnitud del retardo no depende de los coeficientes del filtro con lo que éstos se pueden escoger libremente para modelar la respuesta en amplitud.

El retardo introducido por el filtro es: $\frac{\mathrm{M}}{2 * \mathrm{Frecuencia\ \ muestreo}}$ segundos.

Respuesta de frecuencia para FIR Tipo II:

Teniendo en cuenta que es un filtro de longitud par y respuesta impulsional simétrica alrededor del punto central, tenemos que:

$$\boldsymbol{H(z) = \sum_{n=0}^{M} h_n z^{-n} = h_0 + h_1 z^{-1} + \ldots + h_M z^{-M}}$$

con

$$\boldsymbol{h_n = h_{M-n} \quad para \quad 0 \leq n \leq M}$$

Realizando el cambio de variable $\boldsymbol{z = e^{j\omega}}$

$$\boldsymbol{H(e^{j\omega}) = \sum_{n=0}^{M} h_n e^{-j\omega n}}$$

y aprovechando la propiedad de simetría de los coeficientes, la respuesta en frecuencia resulta ser:

$$\boldsymbol{H(e^{j\omega}) = e^{-j\omega\frac{M}{2}}\left(\sum_{k=1}^{(M+1)/2} b_k \cos[\omega(k-\frac{1}{2})]\right)}$$

Siendo:

$$\boldsymbol{b_k = 2\,h\left[\left(\frac{M+1}{2}\right) - k\right] \quad con \quad k = 1, 2, \ldots, \left(\frac{M+1}{2}\right)}$$

Como:

$$\boldsymbol{H(\omega) = A(\omega)\, e^{j\Phi(\omega)}} \qquad \text{siendo} \qquad \boldsymbol{A(\omega) = \sum_{k=1}^{(M+1)/2} b_k \cos[\omega(k-\frac{1}{2})]}$$

$$\boldsymbol{\Phi(\omega) = -\omega\frac{M}{2}}$$

Comprobamos que

$$\boldsymbol{|H(\omega)| = 0 \quad para\ \omega = \pi}$$

Este tipo No es adecuado para modelar Filtros Paso alto ni Para banda.

Respuesta de frecuencia para FIR Tipo III:

Teniendo en cuenta que es un filtro de longitud impar y respuesta impulsional es antisimétrica alrededor del punto central, tenemos que:

$$\boldsymbol{H(z) = \sum_{n=0}^{M} h_n z^{-n} = h_0 + h_1 z^{-1} + \ldots + h_M z^{-M}}$$

con

$$\boldsymbol{h_n = h_{M-n} \quad para \quad 0 \leq n \leq M}$$

Realizando el cambio de variable $\boldsymbol{z = e^{j\omega}}$

$$\boldsymbol{H(e^{j\omega}) = \sum_{n=0}^{M} h_n e^{-j\omega n}}$$

y aprovechando la propiedad de simetría de los coeficientes, la respuesta en frecuencia resulta ser:

$$\boldsymbol{H(e^{j\omega}) = j e^{-j\omega\frac{M}{2}} \left(\sum_{k=1}^{M/2} c_k \, sin(\omega k) \right)}$$

Siendo:

$$\boldsymbol{c_k = 2\, h\left[\left(\frac{M}{2}\right) - k\right] \quad con \quad k = 1, 2, \ldots, \left(\frac{M}{2}\right)}$$

Como:

$$\boldsymbol{H(\omega) = A(\omega)\, e^{j\Phi(\omega)}} \qquad \text{siendo} \qquad \boldsymbol{A(\omega) = \sum_{k=1}^{M/2} c_k \, sin(\omega k)}$$

$$\boldsymbol{\Phi(\omega) = \frac{\pi}{2} - \omega\frac{M}{2}}$$

Y como:

$$\boldsymbol{|H(\omega)| = 0} \quad \text{para} \quad \begin{matrix} \boldsymbol{\omega = 0} \\ \boldsymbol{\omega = \pi} \end{matrix}$$

Este tipo No es adecuado para modelar filtro Filtros Paso bajo ni Paso alto.

Se utilizan para diseñar transformadores de Hilbert (es un tipo de filtro pasa todo que produce un desfase de $\boldsymbol{\pi/2}$ a la señal de entrada) y diferenciadores (determinan la derivada de la señal de entrada).

Respuesta de frecuencia para FIR Tipo IV:

Teniendo en cuenta que es un filtro de longitud par y respuesta impulsional es antisimétrica alrededor del punto central, tenemos que:

$$\boldsymbol{H(z) = \sum_{n=0}^{M} h_n z^{-n} = h_0 + h_1 z^{-1} + \ldots + h_M z^{-M}}$$

con

$$\boldsymbol{h_n = -h_{M-n} \quad para \quad 0 \leq n \leq M}$$

Realizando el cambio de variable $\boldsymbol{z = e^{j\omega}}$

$$\boldsymbol{H(e^{j\omega}) = \sum_{n=0}^{M} h_n e^{-j\omega n}}$$

y aprovechando la propiedad de simetría de los coeficientes, la respuesta en frecuencia resulta ser:

$$\boldsymbol{H(e^{j\omega}) = je^{-j\omega\frac{M}{2}}\left(\sum_{k=1}^{(M+1)/2} d_k \, sin[\omega(k-\frac{1}{2})]\right)}$$

Siendo:

$$\boldsymbol{d_k = 2h\left[\left(\frac{M+1}{2}\right) - k\right] \quad con \quad k = 1, 2, \ldots, \left(\frac{M+1}{2}\right)}$$

Como:

$$\boldsymbol{H(\omega) = A(\omega)\, e^{j\Phi(\omega)}} \qquad \text{siendo} \qquad \boldsymbol{A(\omega) = \sum_{k=1}^{(M+1)/2} d_k \, sin[\omega\left(k-\frac{1}{2}\right)]}$$

$$\boldsymbol{\Phi(\omega) = \frac{3\pi}{2} - \omega\frac{M}{2}}$$

Y como:

$$\boldsymbol{|H(\omega)| = 0} \quad \text{para } \boldsymbol{\omega = 0}$$

Al igual que el tipo III, este filtro solo se utiliza para diseñar transformadores de Hilbert y diferenciadores.

La fase indicada en los dos casos anteriores considera que ***A(ω)*** es positiva, si fuese negativa incluiríamos un retardo adicional de **π** radianes.

10.4 Ventanas de Truncamiento o Enventanado: Características.

Son numerosas las funciones planteadas para enventanar la respuesta impulsional ideal y el decidirse por una u otra depende de las características de nuestro problema, es decir, si dada una longitud del filtro, estamos más interesados en reducir al máximo la zona de transición, atenuar lo más posible los lóbulos secundarios u optar por otras soluciones de compromiso.

Tal y como se comentó anteriormente, la ventana debe implementar la siguiente acción:

$$w(n) = \begin{matrix} funcion\ simétrica & para\ 0 \le n \le M \\ 0 & en\ el\ resto\ situaciones \end{matrix}$$

La expresión matemática de alguna de las principales ventanas se recoge en la tabla 10.1:

Tabla 10.1 Tipos de ventana

Tipo de ventana	$\mathbf{0} \le \boldsymbol{n} \le \boldsymbol{M}$	$\boldsymbol{n} > M$
Rectangular	1	0
Bartlett (triangular)	$\frac{2n}{M} \quad para\ 0 \le n \le \frac{M}{2}$ $2 - \frac{2n}{M} \quad para\ \frac{M}{2} \le n \le M$	0 0
Von Hann (hanning)	$\frac{1}{2}\left[1 - \cos\frac{2\pi n}{M}\right]$	0
Hamming	$0{,}54 - 0{,}46\cos\frac{2\pi n}{M}$	0
Blackmann	$0{,}42 - 0{,}5\cos\frac{2\pi n}{M} + 0{,}08\cos\frac{4\pi n}{N}$	0

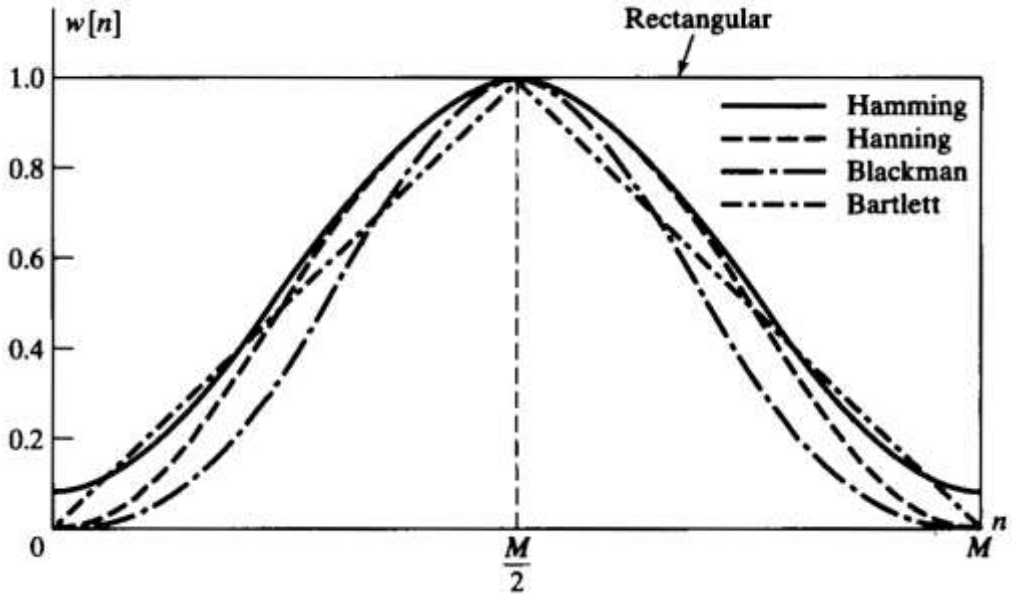


Figura 10.12 Efecto de cada tipo de ventana

Representando para cada tipo de ventana, el valor de $\mathbf{20}\ \boldsymbol{log_{10}}\ \left|\boldsymbol{W}(\boldsymbol{e^{jw}})\right|$ en función de la frecuencia $\boldsymbol{\omega}$, entre 0 y π radianes para un filtro de orden **M** = 50, se obtienen las curvas indicadas en la figura 10.13:

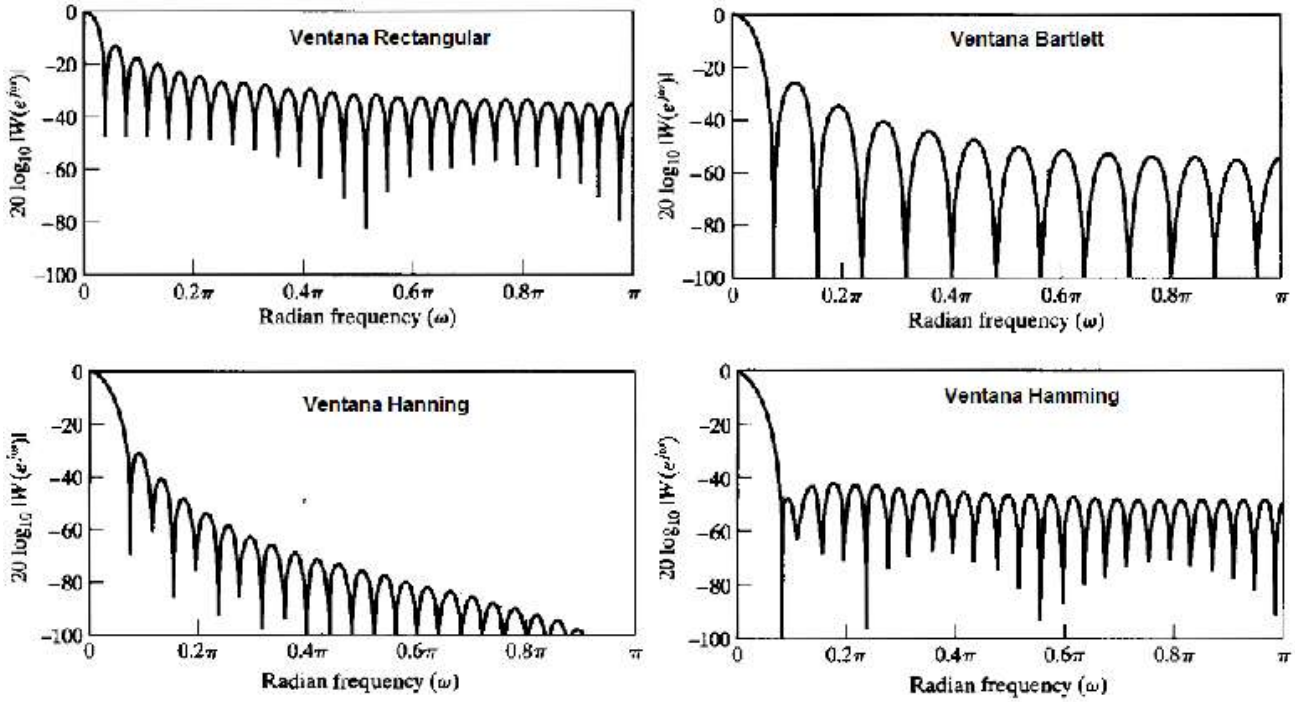


Figura 10.13 Respuesta de cada tipo de ventana

Se define la Anchura del lóbulo principal de la ventana, como el doble del intervalo de frecuencias hasta el primer nulo de los que se producen en las frecuencias

$$\omega_k = \frac{2\pi k}{N} \quad con \;\; k = 1, 2, 3, \dots, N$$

Como vemos, cuando N crece, el lóbulo principal se estrecha.

Los lóbulos secundarios también se estrechan y se atenúan progresivamente, de forma que, en el límite, cuando N tiende a infinito, el lóbulo principal es infinitamente estrecho y los secundarios desaparecen dando lugar a una delta, lo que corresponde con el hecho de que una ventana de longitud infinita es una secuencia de valor constante cuyo contenido espectral es nulo salvo para la componente de continua

La anchura del lóbulo principal está relacionada con la aparición de una banda de transición en el filtro. Cuanto mayor sea el lóbulo principal mayor será la banda de transición del filtro.

La presencia de lóbulos laterales (secundarios) lleva a la aparición de oscilaciones en la respuesta en frecuencia, en ambas bandas, (más apreciables en la banda no pasante).

En la tabla 10.2 se resume algunos de los parámetros característicos de cada una de las ventanas más utilizadas.

Como resumen de este pequeño análisis, podemos tratar de mejorar las prestaciones de un filtro real:

- Aumentando el número de puntos considerados.

  El incremento del orden del filtro eleva su carga computacional.

  Idealmente para M infinito tendríamos una señal de continua, cuyo espectro es un impulso, por lo que al convolucionar obtendríamos la respuesta ideal del filtro).
- Utilizando la ventana que se adecúa mejor a la respuesta impulsional deseada.

Tabla 10.2 Parámetros característicos de cada ventana

Ventana	Anchura lóbulo principal (rad)	Anchura banda de Transición filtro ($\Delta\omega$)	Intensidad lóbulo secundario (dB)
Rectangular	$\frac{4\pi}{M}$	$\frac{1{,}8\pi}{M}$	-13
Bartlett (triangular)	$\frac{8\pi}{M}$	$\frac{6{,}1\pi}{M}$	-25
Von Hann (Hanning)	$\frac{8\pi}{M}$	$\frac{6{,}2\pi}{M}$	-31
Hamming	$\frac{8\pi}{M}$	$\frac{6{,}6\pi}{M}$	-41
Blackmann	$\frac{12\pi}{M}$	$\frac{11\pi}{M}$	-57

10.5 Técnicas para reducir la carga computacional de Filtros Digitales.

Para reducir este coste computacional, se han desarrollado las siguientes estrategias:

Single Constant Multiplication (SCM).

La multiplicación de una variable x por un entero conocido o una constante conocida t, para obtener y = t * x, puede ser realizada en binario mediante:

*) Sumas exclusivas (adds en inglés) y desplazamientos de bits.

o

*) Restas exclusivas (substracts) y desplazamientos de bits.

En este caso se transforman los 0's en desplazamientos, y se restan los 1´s de las constantes más cercanas.

El problema es encontrar la mejor combinación es decir aquella opción que necesita el menor número de operaciones. Esta combinación se conoce con el nombre de **Single Constant Multiplication** (SCM).

Veamos un ejemplo que aclarará la mecánica de cálculo.

La operación $\mathbf{71}\,\boldsymbol{x}$ requiere 3 sumas

$$\mathbf{71}\,\boldsymbol{x} = \mathbf{1000111_2}\,\boldsymbol{x} = \boldsymbol{x} \ll \mathbf{6} + \boldsymbol{x} \ll \mathbf{2} + \boldsymbol{x} \ll \mathbf{1} + \boldsymbol{x}$$

y 4 restas:

$$\mathbf{71}\,\boldsymbol{x} = \mathbf{1000111_2}\,\boldsymbol{x} = (\boldsymbol{x} \ll \mathbf{7} - \boldsymbol{x}) - \boldsymbol{x} \ll \mathbf{5} - \boldsymbol{x} \ll \mathbf{4} - \boldsymbol{x} \ll \mathbf{3}$$

Parecería que por tanto la opción correcta es la que emplea sumas, pero no es así.

Existe otra opción con un coste computacional menor (2 sumas) y que por tanto debe ser la opción a utilizar.

Para determinarla, se hace uso de la llamada Notación [24], [25] **Canonical Signed Digit** (CSD)

Con este nombre se conoce a un sistema de numeración cuyo objetivo es codificar valores de coma flotante mediante el uso de 3 símbolos **{1, 0, y -1}**, en cada uno de los bits del número que se quiere representar, y que se corresponden con el exponente de la potencia de base **2**. El símbolo **-1** también se puede expresar como $\mathbf{\bar{1}}$y como $\mathbf{\underline{1}}$.

En caso de haber varias posibilidades de asignación de bits, se debe elegir la opción de escritura que contenga más ceros.

Así, para la representación del número 71:

2^6	2^5	2^4	2^3	2^2	2^1	2^0	Valor
1	0	0	1	0	0	$\bar{1}$	64+8-1
1	0	1	$\bar{1}$	0	0	$\bar{1}$	64+16-8-1

Escogeremos la representación

$$\mathbf{1001001}\bar{\mathbf{1}}_{\boldsymbol{CSD}} = 64 + 8 - 1 = 71$$

Utilizando la notación CSD, la multiplicación de una variable por un entero conocido o una constante conocida, se puede descomponer en cualquier combinación de Sumas y restas más los consiguientes desplazamientos de bits.

Volviendo al ejemplo indicado anteriormente:

El producto de **71** por x será:

$$\mathbf{1001001}_{\boldsymbol{CSD}}\,\boldsymbol{x} = \;\boldsymbol{x} \ll \mathbf{6} + \boldsymbol{x} \ll \mathbf{3} - \boldsymbol{x}$$

Como vemos, la multiplicación se lleva a cabo con tan solo dos operaciones.

Esta es, por tanto la opción que debe realizarse.

En la figura 10.14, se observa que en general el uso de la notación CSD aporta ventajas sobre la notación binaria, pero aparecen casos en que no es así.

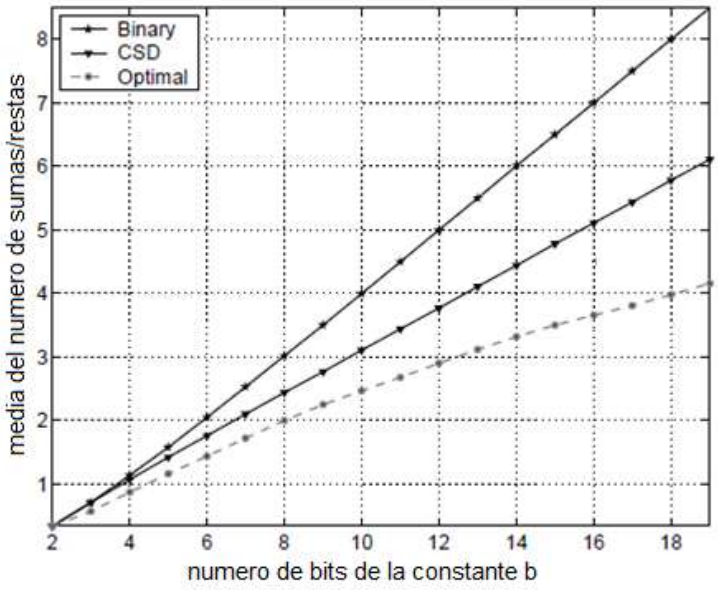


Figura 10.14 Complejidad media según número de bits

En efecto, este es el caso de la multiplicación por 45, representado en la figura 10.15.

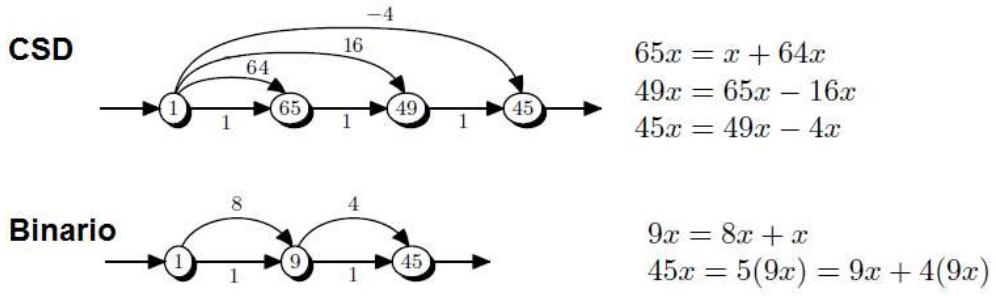


Figura 10.15 Caso en que CSD ofrece resultado más complejo

Multiple Constant Multiplication (MCM).

En la figura 10.16 se representa un Bloque Multiplicador (Multiplier Block), estructura de cálculo para llevar a cabo la multiplicación simultanea de la variable x por las constantes:

$t_0, t_1, t_2, t_3, \ldots, t_n$.

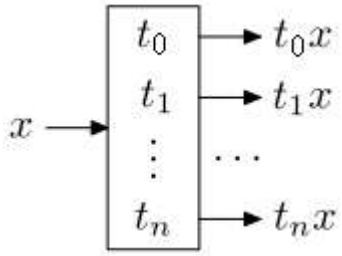


Figura 10.16 Esquema Bloque Multiplicador

Dado que los resultados intermedios de la descomposición de factores de cada una de las multiplicaciones simples a realizar (SCM) pueden ser compartidos en el Bloque multiplicador, serán necesarias muchas menos operaciones, reduciendo por tanto el coste computacional de la operación.

La estrategia que consiste en encontrar la descomposición de factores que implica el menor número de operaciones se conoce con el nombre de Multiple Constant Multiplication (MCM)

En la figura 10.17 se representa el número de operaciones necesarias para llevar a cabo multiplicaciones de hasta 100 constantes de tamaño 12-bits mediante Single Constant Multiplication (SCM) y mediante el algoritmo RAG-n, uno de los algoritmos disponibles de Multiple Constant Multiplication, diseñado por Dempster y Macleod [26] en 1995.

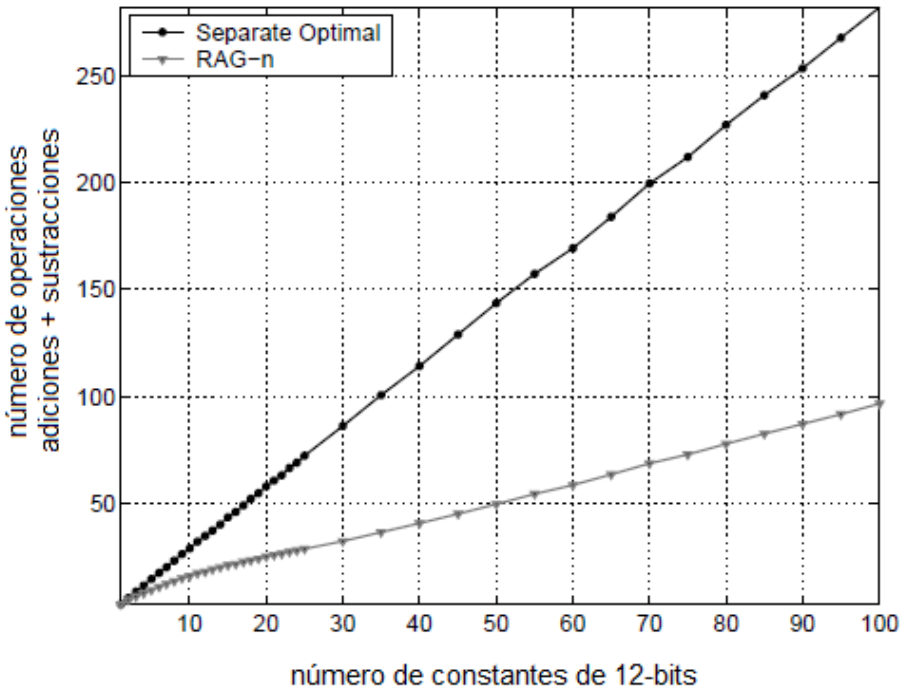


Figura 10.17 Comparativa SCM y RAG-n

10.5.1 Algoritmos existentes para determinar Multiple Constante Multiplication.

En la bibliografía se describen cuatro tipos de algoritmos, con un enfoque distinto para determinar el número óptimo de factores necesarios para llevar a cabo la operación global en el Bloque multiplicador.

Algoritmos Basados en la Recodificación de Bits o Digit-based recoding (DBR).

Los algoritmos Basados en la Recodificación de Bits (Digit-based recoding) se fundamentan en la descomposición directa en factores desde la representación en bits de la constante.

Su ventaja principal es su bajo coste computacional, típicamente lineal frente al número de bits de las constantes a multiplicar.

A este grupo pertenecen la notación binaria y la notación CSD, ya descritas en el apartado anterior.

En 2001, Coleman amplió el campo de trabajo de estos algoritmos, definiendo otro sistema de numeración, llamado **Radix4-CSD** o simplemente **CSD4 [27], [28]**, con mayor capacidad de reducción del número de operaciones.

La numeración CSD4 utiliza la base 4 en lugar de la base 2 del CSD original.

Los exponentes del alfabeto CSD4 son {-15, -13, -11, -9, -7, -5, 0, 5, 7, 9, 11, 13, 15}

Cualquier constante es representada en este tipo de numeración como:

4^3	4^2	4^1	4^0	.	4^{-1}	4^{-2}	4^{-3}	4^{-4}	4^{-5}
x_3	x_2	x_1	x_0	.	x_{-1}	x_{-2}	x_{-3}	x_{-4}	x_{-5}

El punto indica el separador decimal.

Así, el número 27, en notación CSD4 es 000057.

4^5	4^4	4^3	4^2	4^1	4^0	Valor
0	0	0	0	5	7	20+7

Mientras que en notación CSD es: 100101

2^5	2^4	2^3	2^2	2^1	2^0	Valor
1	0	0	1	0	1	32-4-1

Como podemos comprobar, para este ejemplo, la reducción de bits no ceros es del 66% (2 en notación CSD4 frente a 3 en notación CSD).

Algoritmos basados en la Eliminación de Subexpresiones Comunes (Common Subexpression Elimination) (CSE).

Estos algoritmos derivan directamente de los algoritmos DBR.

La idea básica es encontrar pautas comunes en las representaciones de las constantes una vez que éstas se han transformado mediante un sistema de numeración adecuado tal como CSD o CSD4, y posteriormente agruparlas para evitar realizar multiplicaciones repetitivas.

A este grupo pertenecen los métodos de Pasko 1999 [29]; Lefevre 2001 [30] y Hartley 1996 [31].

Sea la Forma traspuesta de un Filtro FIR, como el indicado en la figura 10.18.

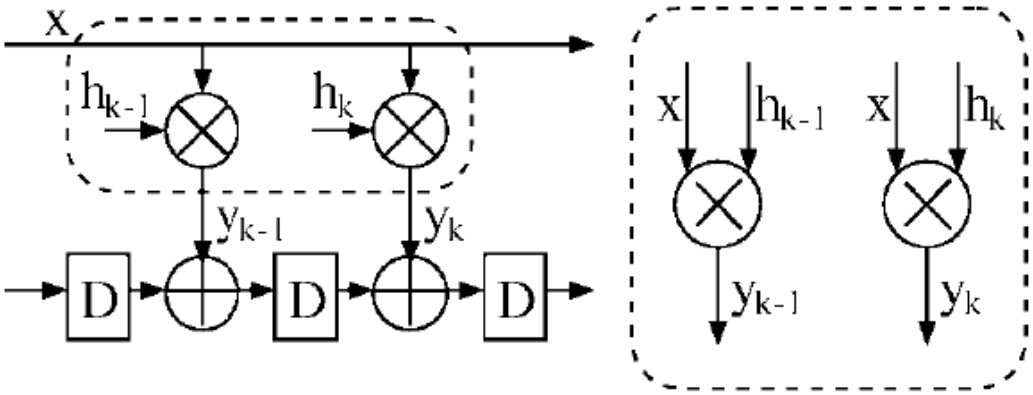


Figura 10.18 Forma traspuesta de un Filtro FIR

Supongamos que las constantes h_k y h_{k-1} son:

$$h_k = 0.0\underline{1}0100$$
$$h_{k-1} = 0.100\underline{1}01$$

El primer paso es la localización de pautas en los coeficientes de las constantes

$yk\text{-}1 = 0.010100_{CSD} \cdot x$	$y_{k-1} = 0.1000\underline{1}01_{CSD} \cdot x$
$= - x >>2 + x >>4$	$= x >>1 - x>>4 + x>>6$
$= (- x + x >>2) >>2$	$= (- x + (- x + x >>2) >>3) >>1$

Pauta Común = -x + x >>2

El segundo paso consiste en aprovechar los cálculos realizados con la pauta para evitar la repetición de multiplicaciones, como se indica en la figura 10.19.

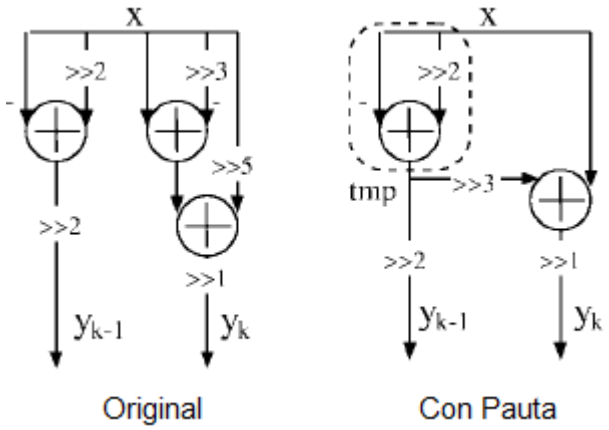


Figura 10.19 Cálculos simplificados con pauta

y_{k-1} = Pauta >>2	y_{k-1} = (- x + Pauta >>3) >>1

La desventaja de estos algoritmos es que el rendimiento depende en gran manera del sistema de numeración utilizado.

Algoritmos basados en Grafos

En este tipo de algoritmos se construye iterativamente un grafo, como el indicado en la figura 10.20, que representa el Bloque Multiplicador.

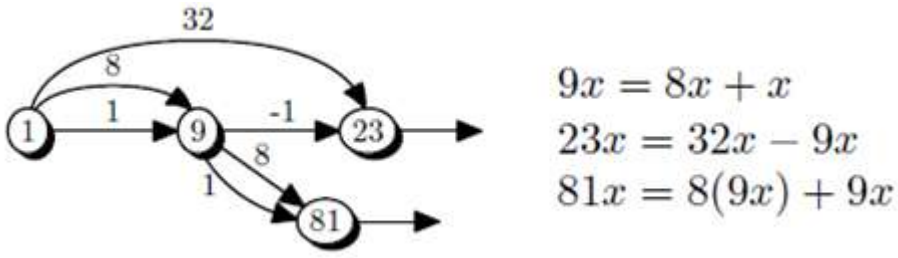


Figura 10.20 Grafo Bloque Multiplicador

La construcción del grafo está guiada por una regla heurística que determina el nuevo vértice que se añadirá al grafo.

Estos algoritmos ofrecen más grados de libertad para no quedar restringidos a una representación particular de los coeficientes, o a una determinada topología como ocurre en los algoritmos Basados en la Recodificación de Bits (DBR) y típicamente producen soluciones con el menor número de operaciones de todos los métodos descritos.

Entre los algoritmos más conocidos se encuentran los siguientes:

- Bernstein 1986 [32]
- Bull and Horrocks 1991 [33]
- Algoritmo RAG-n diseñado por Dempster and Macleod 1995 [34]
- Hcub diseñado por Voronenko y Püschel en 2007 [35], e implementado en el Programa Spiral [36], [37], que será el algoritmo usado en este trabajo.

Algoritmos Hibridos.

Este tipo de algoritmos combina opciones de todas las clases descritas con anterioridad.

Así, por ejemplo, Choo et al. 2004 [38], construye el grafo del Bloque Multiplicador con topología fija para calcular los llamados coeficientes diferenciales y después les une a un algoritmo CSE para llevar a cabo la multiplicación con los coeficientes diferenciales.

10.6 El Proyecto Spiral.

Tal y como se recoge en la página de inicio del Proyecto Spiral, http://spiral.net/, el propósito principal de este grupo es desarrollar software y hardware que permita el acceso libre a algoritmos de Procesamiento Digital de Señales (DSP).

Figura 10.21 Logo de Spiral.net

El proyecto Spiral comprende un equipo interdisciplinar de investigadores que abarcan las áreas de Procesamiento Digital de Señales (DSP), creación de algoritmos de cálculo, compilación, diseño de arquitectura de computadores, etc., con cuyo concurso han conseguido implantar código para el procesamiento de:

- SMP (Multicore)/Vector (código que usa SIMD short vector instructions y multiplethreadings para los chips multiprocesadores más recientes).
- GPUs (Graphical Processing Units).
- Cell processor code.
- Clusters en Distributed memory parallel.
- Verilog for FPGAs.
- Software FPGA acelerado.

Su herramienta principal es el sistema generador de programas SPIRAL, que produce de forma totalmente autónoma, implementaciones hardware para el procesamiento de señales digitales.

Además el proyecto Spiral proporciona los generadores online de acceso libre para el tratamiento de software y hardware, indicados en la tabla 10.3.

Tabla 10.3 Generadores online Spiral

Software (en Lenguaje C)	Hardware (en Lenguaje Verilog)
Transformaciones lineales: DFT Discrete Fourier Transform. DCT Discrete Cosine Transforms.	Discrete Fourier Transform (DFT) Customized sorting networks in synthesizable RTL Verilog
Multiplierless multiplication	Multiplier Blocks Parallel multiple constant multiplication (MCM) Multiplexed Multiple constant multiplication (MUX-MCM)
Viterbi decoder	Filtros Multiplierless

10.7 Verilog.

Verilog es un lenguaje de descripción de hardware (Hardware Description Language, o HDL) usado para modelar sistemas electrónicos. [39]

El lenguaje, algunas veces llamado Verilog HDL, soporta el diseño, prueba e implementación de circuitos analógicos, digitales y de señal mixta a diferentes niveles de abstracción.

Los diseñadores de Verilog querían un lenguaje con una sintaxis similar a la del lenguaje de programación C, de tal manera que le resultara familiar a los ingenieros y así fuera rápidamente aceptada.

El lenguaje tiene un preprocesador como C, y la mayoría de palabras reservadas de control como "if", "while", etc, son similares. El mecanismo de formateo en las rutinas de impresión y en los operadores del lenguaje (y su precedencia) son también similares.

Figura 10.22 Logo de IEEE Verilog

A diferencia del lenguaje C, Verilog usa Begin/End en lugar de llaves para definir un bloque de código. Por otro lado la definición de constantes en Verilog requiere la longitud de bits con su base.

Verilog no tiene estructuras, apuntadores o funciones recursivas.

Finalmente el concepto de tiempo, muy importante en un HDL, no se encuentra en C.

El lenguaje difiere de los lenguajes de programación convencionales, en que la ejecución de las sentencias no es estrictamente lineal.

Un diseño en Verilog consiste de una jerarquía de módulos.

Los módulos son definidos con conjuntos de puertos de entrada, salida y bidireccionales. Internamente un módulo contiene una lista de cables y registros.

Las sentencias concurrentes y secuenciales definen el comportamiento del módulo, describiendo las relaciones entre los puertos, cables y registros.

Las sentencias secuenciales son colocadas dentro de un bloque begin/end y ejecutadas en orden secuencial, pero todas las sentencias concurrentes y todos los bloques begin/end son ejecutadas en paralelo en el diseño.

Un módulo puede contener una o más instancias de otro módulo para definir un sub-comportamiento.

Un subconjunto de sentencias en el lenguaje es sintetizable.

Si los módulos en un diseño contienen sólo sentencias sintetizables, se puede usar software para convertir o sintetizar el diseño en una lista de nodos que describe los componentes básicos y los conectores que deben implementarse en hardware.

La lista de nodos puede entonces ser transformada en una forma describiendo las celdas estándar de un circuito integrado, por ejemplo ASIC, o una cadena de bits para un dispositivo de lógica programable (PLD) como puede ser una FPGA o un CPLD.

10.8 Diseño de Circuitos Integrados de Aplicación Específica.

Con el termino Diseño Circuitos Integrados de Aplicación Específica o **Diseño ASIC** del inglés "Application-Specific Integrated Circuit design" se conoce la técnica utilizada para el diseño, construcción y puesta en funcionamiento de circuitos integrados construidos a la medida para un uso particular y especifico.

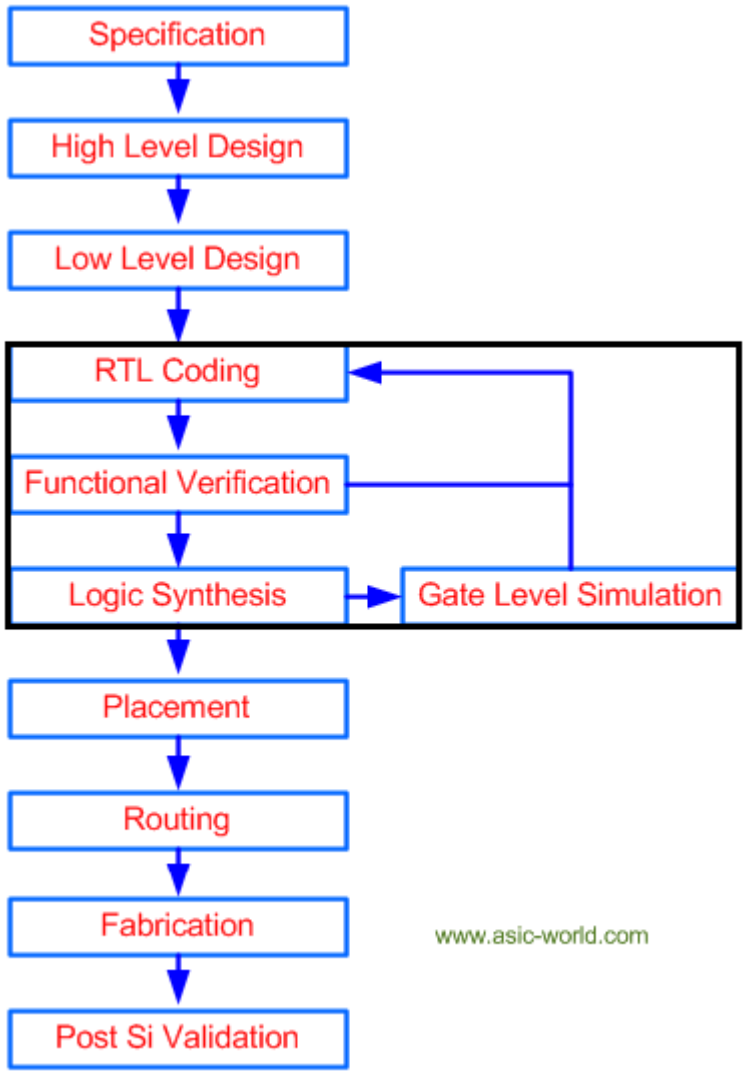


Figura 10.23 Etapas de diseño ASIC

Desde 1985 en que está disponible Verilog, el primer lenguaje HDL (Hardware Description Languaje) [39], el diseño ASIC consta de las etapas indicadas en la figura 10.23:

1) En la etapa de **especificación**, el equipo de diseño completa el análisis de Requerimientos del sistema, conjunto de prestaciones requeridas al nuevo ASIC.
2) En la etapa de **diseño de alto nivel**, se describen los bloques que componen el circuito a diseñar y el modo en que se comunican.
3) ***Low Level Design*** **o Micro Diseño** es la fase en que se describe como se implementará cada bloque. Esta etapa contiene detalles del estado de las maquinas, contadores, multiplexores, decodificadores, registros internos…
4) En la etapa de **codificación RTL** el Micro Diseño es convertido en código Verilog o VHDL, que permite obtener un gran conjunto de elementos de bajo nivel, llamados Celdas Estándares.

   Estos elementos constituyen una colección de puertas precaracterizadas, tales como NOR de 2 entradas, NAND de 2 entradas, inversores, etc., con geometría y tamaño estandarizado, diseñadas de acuerdo con los criterios recogidos en las Lambda Design Rules [40].

   El tamaño final de la puerta viene determinado por la tecnología de transistor utilizada.

   El conjunto de Celdas Estándares más las interconexión entre ellas, constituyen la lista de nodos a nivel de puerta.

   En esta misma etapa, se realiza además la Verificación de las características Funcionales del modelo, modificando si fuese necesario el diseño.

   En conjunto ambas fases representan entre el 60 y el 70% del tiempo total de diseño del circuito ASIC.
5) La lista de nodos a nivel de puerta es importada y colocada de forma que ocupe el menor espacio físico posible para formar el circuito que representa el ASIC.

   Cada una de las puertas debe ser colocada de forma que se consiga minimizar las siguientes variables:

   - Cableado del circuito. No solo minimiza el tamaño del chip y su costo, sino el consumo y el retraso proporcionales a la longitud de cable empleado.

- Timing. Como el ciclo de reloj de un circuito viene determinado por su ruta más larga (ruta crítica). se debe asegurar que ninguna ruta supera el retraso máximo especificado.
- Consumo de energía: El objetivo es aliviar puntos calientes y suavizar los gradientes de temperatura dentro del circuito.

6) En esta etapa, también llamada ***Wire Routing***, se procede a conectar, mediante los correspondientes hilos los componentes colocados en la placa del circuito. El objetivo del Router es asegurar que todos los pines de las celdas estén correctamente conectados, no haya ninguno sin conectar, no se viole ninguna regla de conexión eléctrica y no se produzcan fenómenos de interferencia electromagnética.
7) En la etapa ***Fabrication*** tiene lugar la preparación de las fotomáscaras, con las que el fabricante producirá los circuitos integrados, mediante técnicas de fotolitografía.

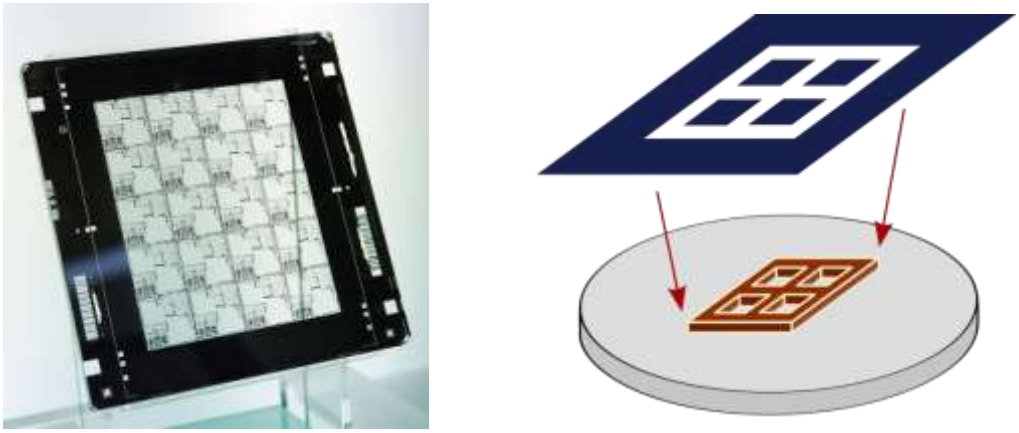


Figura 10.24 Fotomáscaras y proceso de fotolitografia

8) En esta última etapa, **Validación Post Silicio**, el circuito recién fabricado se coloca en el entorno real y se procede a realizar pruebas de funcionamiento. Estas pruebas demostrarán que el dispositivo funcionará en los rangos de temperatura y voltaje extremos.

Reglas de Diseño Lambda para la construcción de circuitos ASIC.

Consideremos el proceso de fabricación de una puerta lógica inversor, simbolizada en la figura 10.25, y cuya sección trasversal se representa en la figura 10.26.

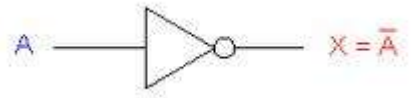


Figura 10.25 Representación simbólica de una puerta inversor

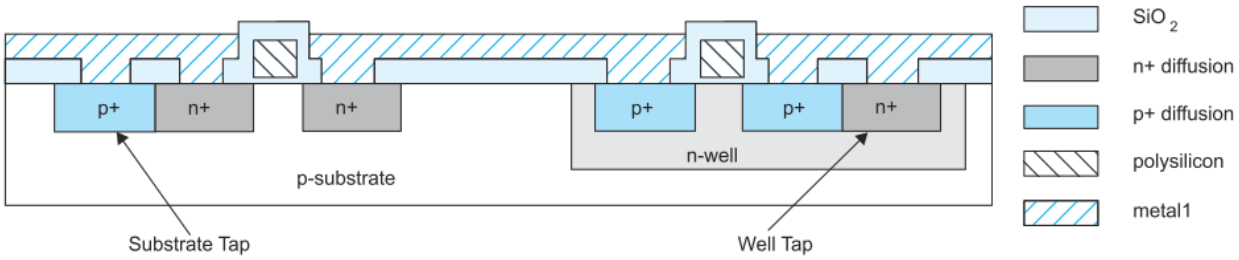


Figura 10.26 Sección transversal de una puerta inversor con contactos E/S

La secuencia de fabricación consiste en una serie de etapas en las que, de forma ordenada, y mediante un proceso de litografía se van depositando cada una de las capas (mascaras) que constituyen el chip y que en este caso concreto son: n- well, polisilicon, n+ diffusion, p+ diffusion, contactos y metal, tal como se indica en la figura 10.27.

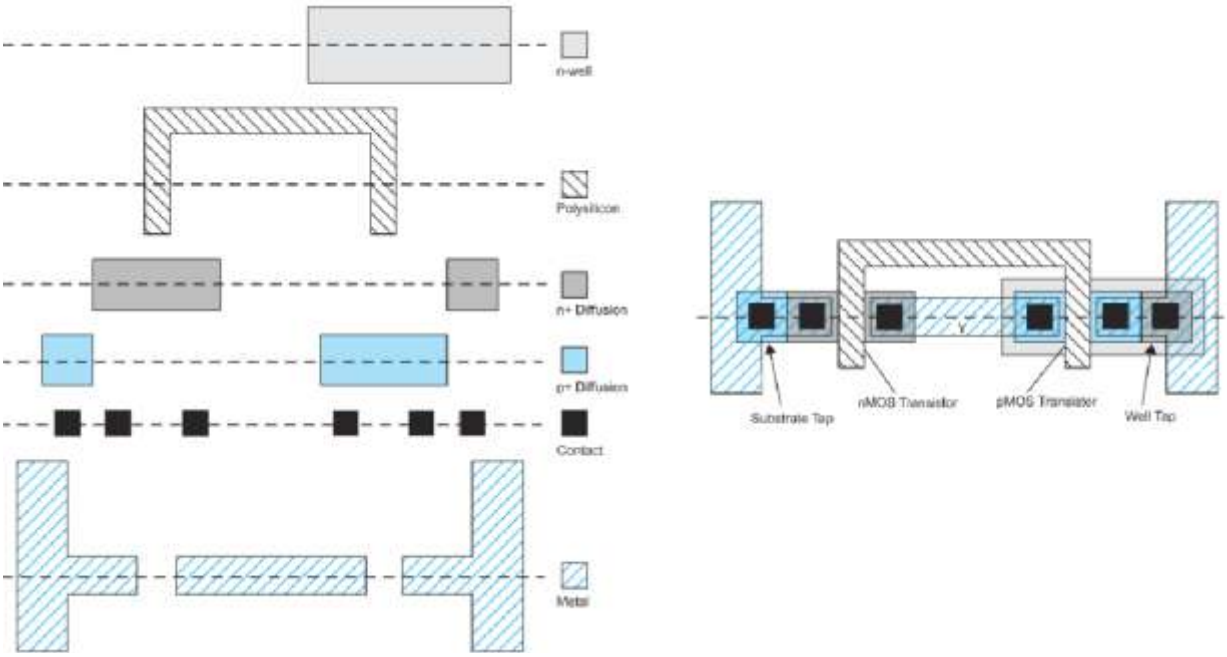


Figura 10.27 Sección transversal de una puerta inversor con contactos E/S

Las reglas de diseño Lamba expuestas por Mead y Conway en 1980 [41], describían los tamaños y distancias a las que deberían ser colocados cada uno de los elementos que constituían la puerta lógica, basando la resolución grafica del proceso fotolitográfico en el parámetro Lambda, λ.

Generalmente λ es definido como la mitad de la longitud mínima del eje longitudinal L de un transistor de la pertinente tecnología, y cuyo esquema constructivo queda recogido en la figura 10.28.

La ventaja de utilizar el factor Lambda es que ante un cambio de tecnología del transistor, es decir, ante un nuevo valor de L, menor que el correspondiente a la tecnología previa, el proceso fotolitográfico se puede calcular de forma inmediata sin más que utilizar el nuevo factor de proporcionalidad λ.

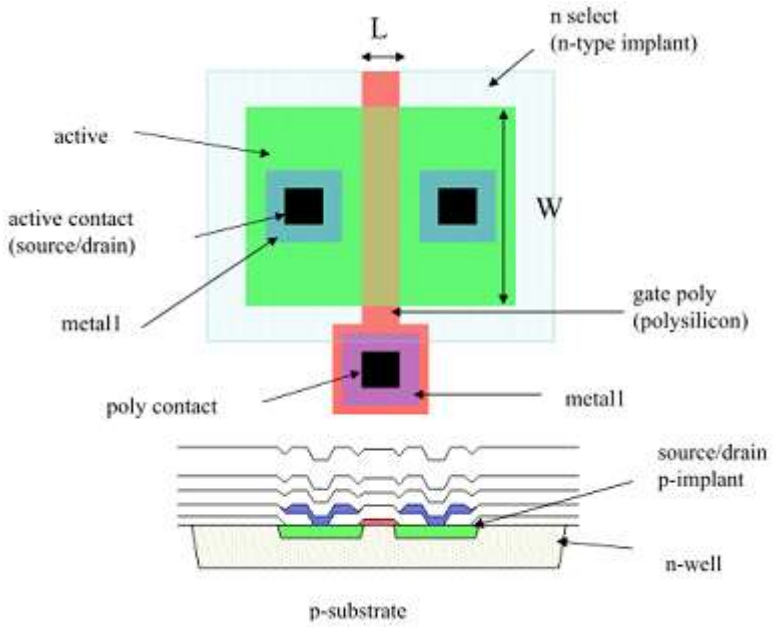


Figura 10.28 Parámetros de diseño de un transistor

En la figura 10.29 se presenta un ejemplo de aplicación de las reglas de diseño Lambda.

En la página web de MOSIS se puede descargar la versión V8 de las reglas actualmente vigentes de diseño Escalable CMOS (SCMOS [42]).

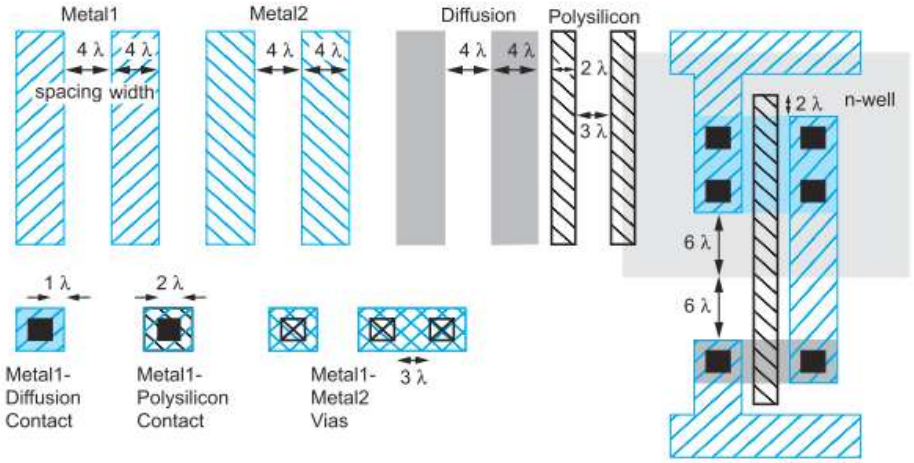


Figura 10.29 Reglas de diseño para colocación de dos capas metálicas sobre n-well

De la misma forma se establecen Reglas de Diseño para conseguir una disposición lo más compacta posible de todos los componentes del circuito, en la idea de ocupar la menor superficie.

Así en la figura 10.30 se recoge la disposición óptima para una celda estándar NAND de tres entradas.

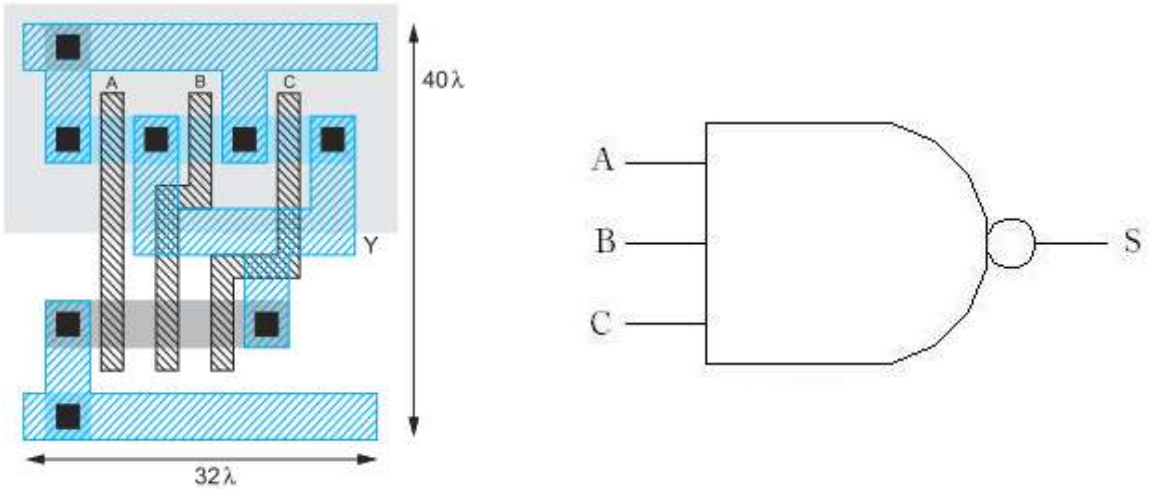


Figura 10.30 Esquema de una puerta 3-input NAND y su esquema lógico

Como puede comprobarse, la puerta ocupa una superficie de 32 λ * 40 λ = 1280 λ^2.

El programa que emplea el HDL utilizado para el diseño del circuito se encarga de recopilar automáticamente la superficie de cada uno de los componentes del circuito. Verilog realiza esta misma función y por tanto se aprovechará los datos aportados para utilizarlos en este trabajo.

10.9 Simulación Digital de errores mediante la herramienta ModelSim.

Para llevar a cabo la simulación digital cualquier circuito ASIC, se puede utilizar la herramienta ModelSim. La versión Student PE 10.1b, edición freeware puesta a disposición de estudiantes e instituciones académicas sin fin lucrativo, puede descargarse en la dirección: http://model.com/content/modelsim-pe-student-edition-hdl-simulation .

Figura 10.31 Logo ModelSim.

Esta edición presenta funcionalidad completa pero con un rendimiento en torno al 30% del rendimiento de la versión PE comercial para el procesado de hasta 10.000 líneas de código, y un valor en torno al 1% cuando el código correspondiente al proyecto a simular supera dicha barrera de líneas.

Al igual que ocurre con otros entornos de desarrollo, ModelSim gestiona la información de cada diseño mediante un proyecto, facilitando de esta forma su desarrollo.

Un proyecto se compone de:

- Un directorio de trabajo en el que se almacenan los distintos ficheros generados durante la compilación y simulación de un diseño.
- Los ficheros fuente VHDL del diseño.
- Las librerías creadas.
- La configuración empleada en el simulador.

Toda esta información se almacena en un fichero con el mismo nombre que el proyecto y la extensión .mpf.

Las operaciones que se pueden realizar con un proyecto son crearlo, abrirlo, cerrarlo o borrarlo. Estas se realizan mediante las opciones New, Open, Close y Delete del menú File.

Creación de un proyecto.

Para crear el proyecto empleado en la simulación funcional del diseño del ejemplo se selecciona la opción File => New => Project de la barra de menús.

En ese momento se muestra en la pantalla la ventana Create Project, donde se configuran las opciones particulares escogidas, tal y como se muestra en la figura 10.32.

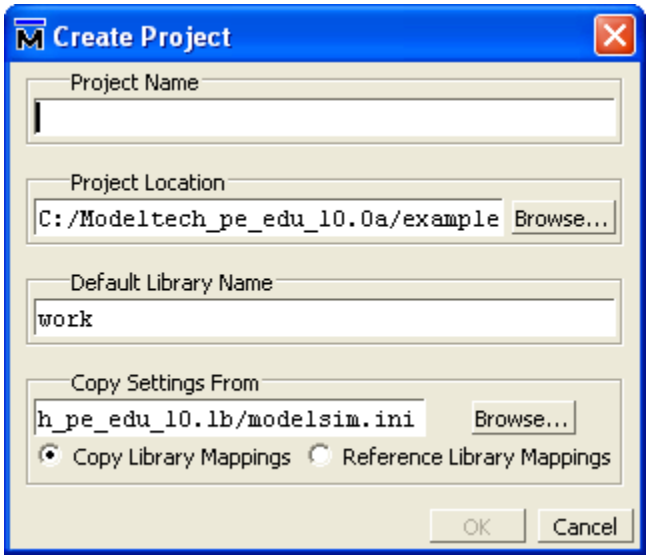


Figura 10.32 Creación proyecto en ModelSim.

Acto seguido, aparece la ventana "Add ítems to the Project", recogida en la figura 10.33. Seleccionamos la opción "Add Existing File".

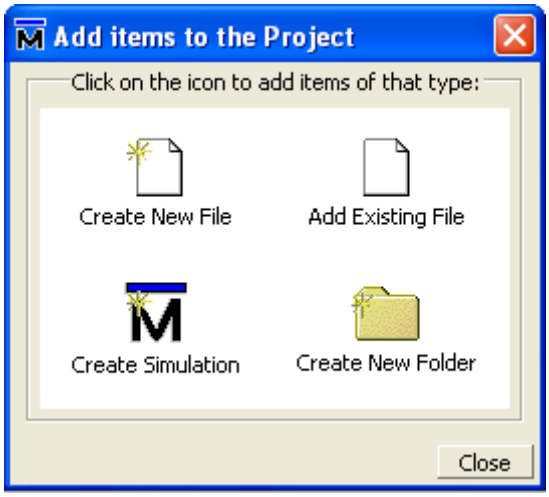


Figura 10.33 Añadiendo elementos al proyecto en ModelSim.

El sistema solicita el nombre y ubicación del fichero que contiene el código de la simulación a realizar. Seleccionaremos el archivo Verilog generado por DetectError.

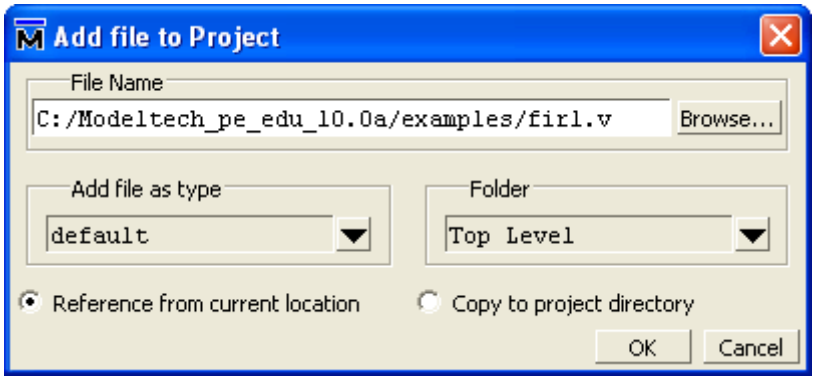


Figura 10.34 Añadiendo archivo Verilog en ModelSim.

El programa presenta el código fuente del archivo Verilog en la pantalla, tal y como se indica en la figura 10.35, estando listo para la siguiente etapa del proceso.

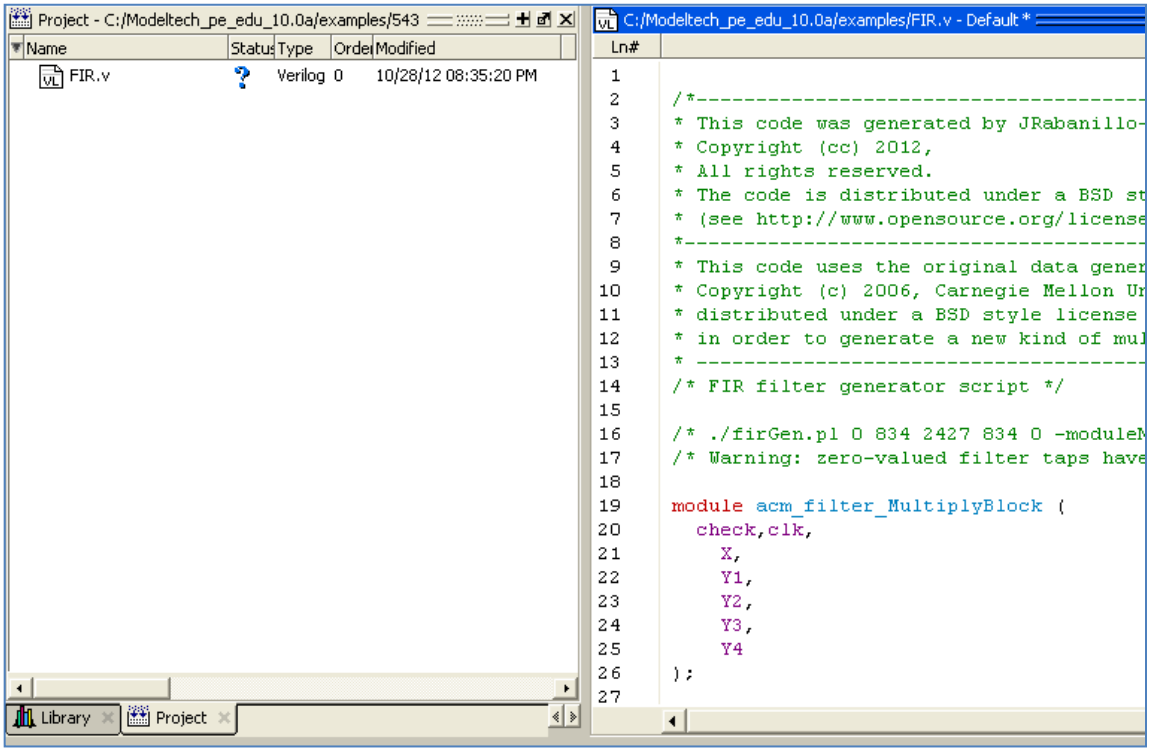


Figura 10.35 Edición de archivo Verilog en ModelSim.

Compilación del proyecto.

Para poder llevar a cabo la simulación del proyecto, se debe compilar. Para ello se elije la opción “Compile” del menú “Compile”, indicado en la figura 10.36.

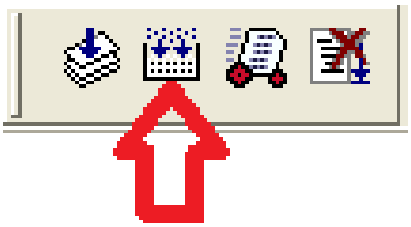


Figura 10.36 Compilando proyecto en ModelSim.

Durante la compilación el simulador muestra en la consola la información referente al desarrollo de la misma. Si aparecen errores, se muestran en color rojo sus textos explicativos.

Haciendo doble click sobre la línea de error se muestra automáticamente la línea del código que produjo dicho error en la ventana de código fuente.

Si no se produce ningún error durante el proceso de compilación, el sistema presenta un mensaje que confirma dicha situación, permitiendo iniciar la propia simulación.

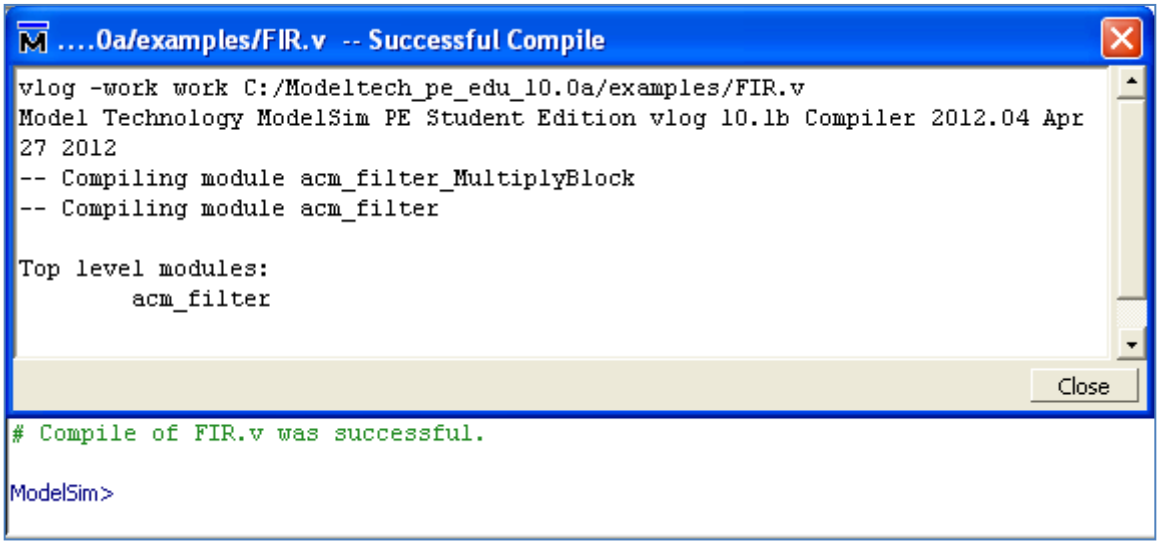


Figura 10.37 Compilación satisfactoria del proyecto en ModelSim.

Simulación del proyecto.

Por último, para preparar el proyecto para la inserción de fallos, debemos simular el Filtro FIR representado en el archivo Verilog de nuestro proyecto.

Para ello se debe pulsar el icono indicado con la flecha en rojo, opción "Simulate" del menú "Compile" situado en la barra de herramientas

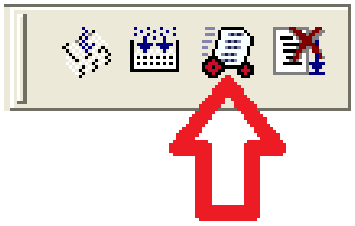


Figura 10.38 Simulando el proyecto en ModelSim.

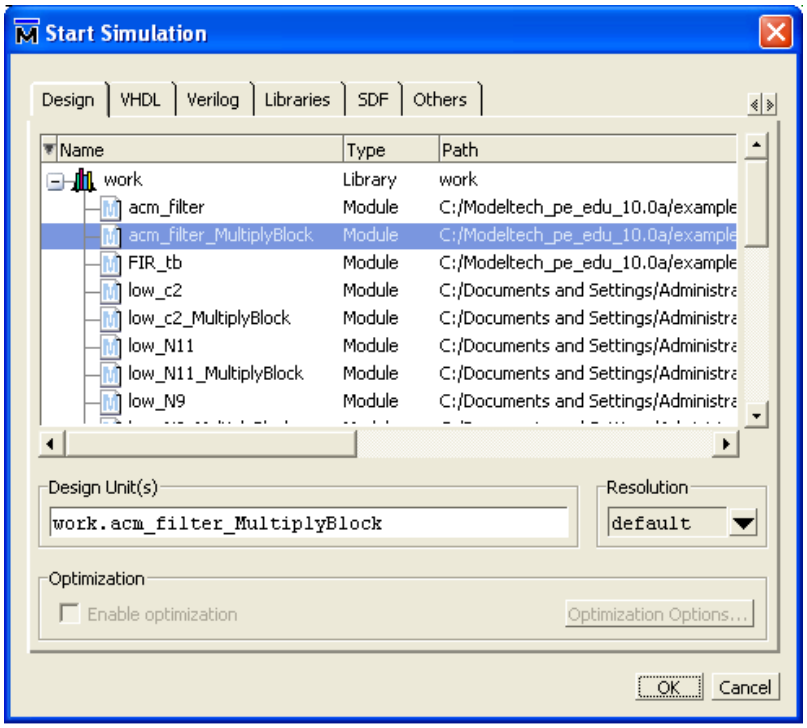


Figura 10.39 Simulando el Bloque Multiplicador en ModelSim.

Tras la confirmación oportuna, el proyecto está listo para la siguiente etapa del proceso de simulación.

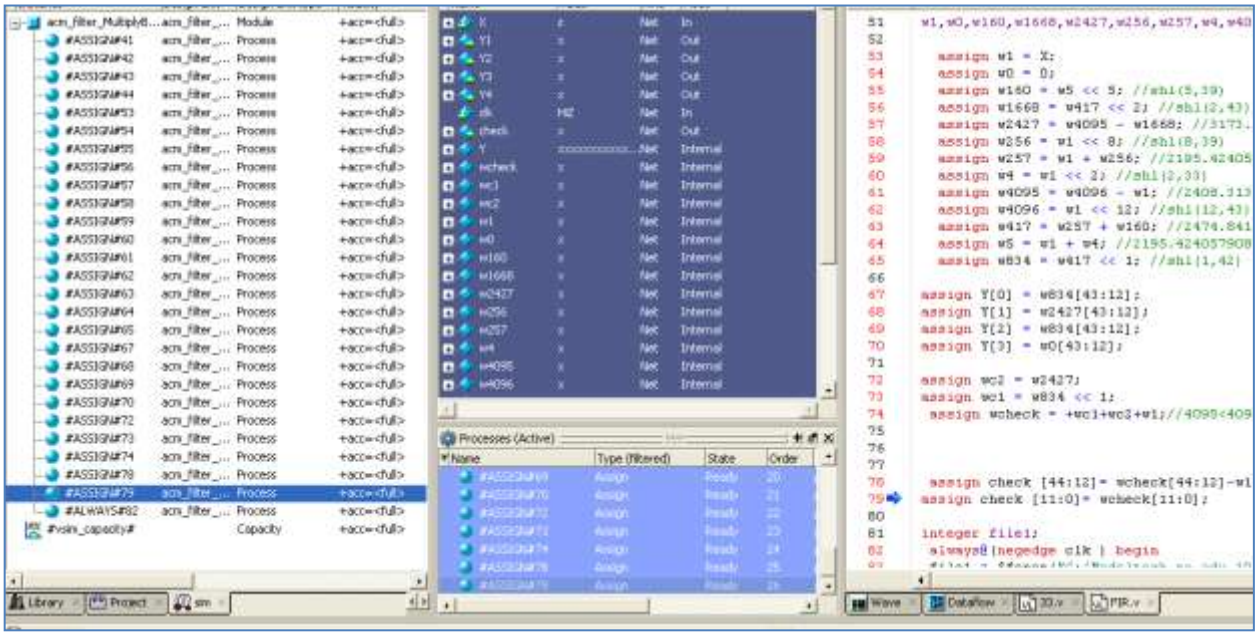


Figura 10.40 Simulación iniciada en ModelSim.

Referencias Citas.

[1] Victor Franz Hess The Nobel Prize in Physics 1936 biography. http://hess.nobelpr.com/1.htm.

[2] Observatorio Pierre Auger. http://auger.org.ar/

[3] Centaurus A bajo una mirada más profunda. http://www.eso.org/public/spain/news/eso1221/

[4] Agujero negro. http://www.nasa.gov/audience/forstudents/k-4/stories/what-is-a-black-hole-k4.html

[5] Fan Wang, "Soft Error Rate Determination for Nanometer CMOS VLSI Circuits" Thesis Doctoral. Auburn University, Alabama USA, 2008 http://www.eng.auburn.edu/~agrawvd/THESIS/WANG/FanWangMS_Thesis.pdf.

[6] T. C. May and M. H. Woods. "Alpha-particle-induced soft errors in dynamic memories." IEEE Trans. on Electron Devices, 26(1):2–9, 1979

[7] J.F.Ziegler and W. A. Lanford. "The effect of Cosmic Rays on computer memories". Science, vol 206, 1979

[8] J.F.Ziegler "Terrestrial cosmic rays". IBM J. of Research and Development, 40(1):19–39, 1996

[9] Shubhendu S. Mukherjee, Joel Emer and Steven K. Reinhardt. "The Soft Error Problem: An Architectural Perspective" 11th International Symposium on High-Performance Computer Architecture, pp 243-247, 2005. http://web.eecs.umich.edu/~stever/pubs/hpca05-ser.pdf

[10] http://spiral.ece.cmu.edu/mcm/gen.html

[11] dotty(1) - Linux man page http://linux.die.net/man/1/dotty

[12] http://www.graphviz.org/pdf/dotguide.pdf

[13] http://www.graphviz.org/pdf/leftyguide.pdf

[14] ModelSim - Advanced Simulation and Debugging http://model.com/

[15] Tcl http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html

[16] M. S. Gordon, P. Goldhagen, K. P. Rodbell, T. H. Zabel, H. H. K. Tang, J. M. Clem, and P. Bailey, "Measurement of the Flux and Energy Spectrum of Cosmic-Ray Induced Neutrons on the Ground," IEEE Transactions on Nuclear Science, vol. 51, no. 6, pp. 3427-3434, Dec. 2004

[17] Fernanda Gusmão de Lima. “Single Event Upset Mitigation Techniques for Programmable Devices” Trabajo de graduación EQ-02 PPGC-UFRGS. Universidad Federal de Rio Grande del Sur. 2000. Disponible en http://www.inf.ufrgs.br/~fglima/lima_qualifying.pdf

[18] IBM. SOI Technology: IBM’s Next Advance in Chip Design. In: http://www.ibm.com (Jan. 2000).

[19] WEAVER, H.; et al. An SEU Tolerant Memory Cell Derived from Fundamental Studies of SEU Mechanisms in SRAM. In: IEEE Transactions on Nuclear Science. Vol. 34, Nº 6, December 1987

[20] RABAEY, Jan. “Digital Integrated Circuits - A Design Perspective”. Upper Saddle River : Prentice Hall, 1996. 702 p.

[21] WHITAKER, S.; CANARIS, J.; LIU, K. SEU Hardened Memory Cells for CCSDS REED Solomon Encoder. In: IEEE Transactions on Nuclear Science. Vol. 38, Nº 6, December, 1991

[22] P. Sweeny, Error Control Coding, From Theory to Practice, Wiley, 2002

[23] Robert C. Baumann, Fellow, IEEE "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies" IEEE TRANSACTIONS ON DEVICE AND MATERIALS RELIABILITY, VOL. 5, NO. 3, SEPTEMBER 2005

[24] Heechul Kim Lewis “Canonical Signed Digit Study Part II: FIR Digital Filter Simulation Results” Research Center NASA Technical Memorandum 107335 October 1996

[25] J. O. Coleman and A. Yurdakul, “Fractions in the Canonical-Signed-Digit Number System,” in Proc. 2001 Conf. on Information Sciences and Systems, (Johns Hopkins University), Mar. 2001.

[26] Dempster, A. G. and Macleod, M. D. 1995. Use of minimum-adder multiplier blocks in FIR digital filters. IEEE Transactions in Circuits and Systems-II: Analog and Digital Signal Processing 42, 9, 569–577.

[27] J. O. Coleman “Cascaded coefficient number systems lead to FIR filters of striking computational efficiency”. 2001 Int’l IEEE Conf. on Electronics, Circuits, and Systems (ICECS). Malta, September 2–5, 2001

[28] J. O. Coleman “Express Coefficients in 13-ary, Radix-4 CSD to Create Computationally Efficient Multiplierless FIR Filters,” The 15th European Conf. on Circuit Theory and Design (ECCTD ’01). Espoo, Finland, August 28–31, 2001

[29] Pasko, R., Schaumont, P., Derudder, V., Vernalde, S., and Durackova, D. A new algorithm for elimination of common subexpressions. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18, 1, 58–68, 1999

[30] Lefevre, V. 2001. Multiplication by an integer constant. Tech. rep., INRIA.

[31] Hartley, R. I. 1996. Subexpression sharing in filters using canonic signed digit multipliers. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 43, 10, 677–688.

[32] Bernstein, R. L.. Multiplication by integer constants. Software – Practice and Experience 16, 7, 641–652, 1986

[33] Bull, D. R. and Horrocks, D. H. Primitive operator digital filters. IEE Proceedings G 138, 3, 401–412, 1991.

[34] Dempster, A. G. and Macleod, M. D. 1995. Use of minimum-adder multiplier blocks in FIR digital filters. IEEE Transactions in Circuits and Systems-II: Analog and Digital Signal Processing 42, 9, 569–577.

[35] Voronenko, Y and Püschel, M. "Multiplierless Multiple Constant Multiplication. ACM Transactions on Algorithms", Vol. 3, No. 2, 2007

[36] Voronenko, Yevgen; Franchetti, Franz; de Mesmay, Frédéric; Püschel, Markus. System demonstration of Spiral: Generator for high-performance linear transform libraries. In Algebraic Methodology and Software Technology (AMAST),2008b. 3

[37] Voronenko, Yevgen; Franchetti, Franz; de Mesmay, Frédéric; Püschel, Markus. Generating high-performance general size linear transform libraries using Spiral. In High Performance Embedded Computing (HPEC), 2008a. 68, 69, 112

[38] Choo, H., Muhammad, K., and Roy, K. 2004. Complexity reduction of digital filters using shift inclusive differential coefficients. IEEE Transactions on Signal Processing 52, 6, 1760–1772.

[39] http://www.verilog.com/IEEEVerilog.html

[40] Lynn Conway's VLSI Archive: Main Links http://ai.eecs.umich.edu/people/conway/VLSI/VLSIarchive.mainlinks.html

[41] "CMOS VLSI Design, A Circuits and Systems Perspective", Neil H. E. Weste and David Harris, chapter 1, ene-2004

[42] "MOSIS Escalable CMOS (SCMOS) Design Rules" Revision 8.00, Mayo 11, 2009 http://www.mosis.com/files/scmos/scmos.pdf

43] Protection against soft errors in the space environment: A ?nite impulse response(FIR) ?lter case study J.A. Maestro, P. Reviriego, P. Reyes, O. Ruano, INTEGRATION, the VLSI journal 42 (2009) 128– 136

[44] A. Oppenheim "Discrete time signal processing" 1998, Prentice Hall

[45] http://www-sigproc.eng.cam.ac.uk/~op205/3F3_4_Basics_of_Digital_FIlters.pdf

[46] http://ocw.uv.es/ingenieria-y-arquitectura/filtros-digitales/temario/

[47] "Time-Multiplexed Multiple-Constant Multiplication Peter Tummeltshammer, Student Member, IEEE, James C. Hoe, Member, IEEE, and Markus Puschel, ¨ Senior Member, IEEE"

[48] Universidad de Glasgow 2010
http://www.findpdf.us/search?q=site:wwwi.elec.gla.ac.uk&p=4

Referencias Figuras.

Figura:1.1 ref: Elaboración propia

Figura:1.2 ref: http://apod.nasa.gov/apod/ap060701.html
http://www.physics.hku.hk/~nature/notes/lectures/chap17.html

Figura:1.3 ref: Elaboración propia a partir de http://www.gae.ucm.es/fisica/report.html

Figura:1.4 ref: [6], Página: 22, fig: 2.5

Figura:1.5 ref: [17], Página: 307, fig: 2

Figura:1.6 ref: [17], Página: 307, fig: 2

Figura:1.7 ref: [17], Página: 307, fig: 2

Figura:1.8 ref: [10], Página: 1, fig: 1

Figura:1.9 ref: Elaboración propia

Figura:1.10 ref: Elaboración propia a partir de [44], Página: 355, fig: 6.14

Figura:1.11 ref: Elaboración propia a partir de [44], Página: 342, fig: 6.1

Figura:1.12 ref: Elaboración propia a partir de [44], Página: 367, fig: 6.31

Figura:1.13 ref: Elaboración propia a partir de [44], Página: 367, fig: 6.32

Figura:1.14 ref: Elaboración propia a partir de [44], Página: 327, fig: 5.41-2

Figura:1.15 ref: Elaboración propia a partir de [44], Página: 327, fig: 5.41-2

Figura:1.16 ref: Elaboración propia a partir de [45]

Figura:1.17 ref: Elaboración propia a partir de [45]

Figura:1.18 ref: [46] tema 3, Página: 3.8, fig:

Figura:1.19 ref: http://es.wikipedia.org/wiki/Dise%C3%B1o_de_Filtros_de_Respuesta _Finita_al_Impulso contrastada con [19]

Figura:1.20 ref: http://es.wikipedia.org/wiki/Dise%C3%B1o_de_Filtros_de_Respuesta _Finita_al_Impulso contrastada con [19]

Figura:1.21 ref: http://es.wikipedia.org/wiki/Dise%C3%B1o_de_Filtros_de_Respuesta _Finita_al_Impulso contrastada con [19]

Figura:1.22 ref: http://es.wikipedia.org/wiki/Dise%C3%B1o_de_Filtros_de_Respuesta _Finita_al_Impulso contrastada con [19]

Figura:1.23 ref: http://www.mathworks.es, Página: logo

Figura:1.24 ref: Captura de pantalla Fdatool

Figura:1.25 ref: Captura de pantalla MATLAB

Figura:1.26 ref: Elaboración propia a partir de [44], Página: 367, fig: 6.32

Figura:1.27 ref: Elaboración propia

Figura:1.28 ref: Elaboración propia

Figura:1.29 ref: Captura de pantalla Spiral

Figura:1.30 ref: Grafo generado con código DOT por DetectError

Figura:1.31 ref: Grafo generado con código DOT por DetectError

Figura:1.32 ref: Captura de pantalla grafo Spiral para ejemplo fig 1.51

Figura:2.1 ref: Captura de pantalla grafo Spiral para ejemplo fig 1.51

Figura:2.2 ref: Tema 4 "Bloques Combinacionales" Sistemas Digitales, Prf O.Ruano, Univ. Nebrija, Página: 11, fig:

Figura:2.3 ref: Elaboración propia a partir de http://www.comoustedyasabe.com.ar/datos/Segundo/2do_cuatrimestre/Electronica_Digital/Apuntes_Cuys/Implementacion_de_puertas_basicas_con_transistores(RTL).pdf

Figura:2.4 ref: Elaboración propia

Figura:2.5 ref: Elaboración propia

Figura:2.6 ref: Elaboración propia

Figuras Capitulo 3: ref: Elaboración propia

Figuras Capitulo 4: ref: Elaboración propia excepto

Figura:4.1 ref: Tema 4 "Bloques Combinacionales" Sistemas Digitales, Prf O.Ruano, Univ. Nebrija

Figura:4.2 ref: Elaboración propia a partir de Captura de pantalla grafo Spiral para ej. fig 1.51

Figuras Capitulo 5: ref: Elaboración propia

Figura:10.1 ref: [4], Página: 58, fig: A.2.1

Figura:10.2 ref: http://www.seutest.com/cgi-bin/FluxCalculator.cgi

Figura:10.3 ref: Elaboración propia a partir de [17], Página: 38, fig: 5.1

Figura:10.4 ref: [17], Página: 40, fig: 5.3

Figura:10.5 ref: [17], Página: 40, fig: 5.4

Figura:10.6 ref: Elaboración propia a partir de [17], Página: 48, fig: 6.3

Figura:10.7 ref: Elaboración propia a partir de [17], Página: 51, fig: 6.5

Figura:10.8 ref: [18], Página: 130, fig: 2

Figura:10.9 ref: [18], Página: 130, fig:

Figura:10.10 ref: [18], Página: 130, fig:

Figura:10.11 ref: [44], Página: 368 fig: 6.33

Figura:10.12 ref: [44], Página: 469, fig: 7.21

Figura:10.13 ref: [44], Página: 470, fig: 7.22

Figura:10.14 ref: Elaboración propia a partir de [35], Página: 3, fig: 1

Figura:10.15 ref: [35], Página: 3, fig: 1

Figura:10.16 ref: Elaboración propia a partir de [35], Página: 4, fig: 3

Figura:10.17 ref: [35], Página: 4, fig: 4

Figura:10.18 ref: [29], Página: 58, fig: 1

Figura:10.19 ref: [29], Página: 58, fig: 1

Figura:10.20 ref: [35], Página: 5, fig: 5

Figura:10.21 ref: http://spiral.net/, Página: logo

Figura:10.22 ref: http://www.verilog.com/IEEEVerilog.html, Página: logo

Figura:10.23 ref: Elaboración propia a partir de http://asic-world.com/verilog/design_flow10.html#Introduction

Figura:10.24 ref: http://en.wikipedia.org/wiki/File:Semiconductor_photomask.jpg http://en.wikipedia.org/wiki/File:Mask_illustration.svg

Figura:10.25 ref: Elaboración propia

Figura:10.26 ref: Elaboración propia a partir de [47], Página: 24, fig: 10.33

Figura:10.27 ref: Elaboración propia a partir de [47], Página: 26, fig: 10.35

Figura:10.28 ref: [48] Tema 5, Página: 2, fig: 2

Figura:10.29 ref: [47], Página: 31, fig: 10.39

Figura:10.30 ref: Elaboración propia a partir de [47], Página: 33, fig: 10.42

Figura:10.31 ref: Logo ModelSim

Figura:10.32 ref: Captura de pantalla ModelSim

Figura:10.33 ref: Captura de pantalla ModelSim

Figura:10.34 ref: Captura de pantalla ModelSim

Figura:10.35 ref: Captura de pantalla ModelSim

Figura:10.36 ref: Elaboración propia a partir de Captura de pantalla ModelSim

Figura:10.37 ref: Captura de pantalla ModelSim

Figura:10.38 ref: Elaboración propia a partir de Captura de pantalla ModelSim

Figura:10.39 ref: Captura de pantalla ModelSim

Figura:10.40 ref: Captura de pantalla ModelSim